MODULARIZED FILTERING FOR NAVIGATION

A Thesis
Presented to
The Academic Faculty

By

Dominic J. Macchiaroli

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2017

MODULARIZED FILTERING FOR NAVIGATION

Approved by:

Dr. Gisele Bennett, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Edward Coyle
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Ayanna Howard
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Eric Johnson
School of Aerospace Engineering
Georgia Institute of Technology

Date Approved: April 27, 2017

ACKNOWLEDGEMENTS

The work completed for this thesis would not have been possible without the contributions of a number of individuals. I would like to first thank my advisor, Dr. Gisele Bennett, for her guidance, support, and encouragement throughout this process. Her academic and professional advice have been invaluable. I would like to thank Dr. Eric Johnson for his wonderful introductory course on optimal state estimation and for sharing his insight during the development of this thesis.

I would also like to thank my colleagues at the Electro-Optical Systems Laboratory (EOSL) for all of their advice and support over the last year and a half. They have greatly contributed to my learning experience at Georgia Tech. I would like to specifically thank Dr. Chris Valenta, Dr. Grady Tuell, and Dr. Domenic Carr for providing the opportunity to pursue research in navigation while working for EOSL. Their guidance and assistance have greatly contributed to my professional development. I would also like to thank Ginny Myers for her assistance in binding initial drafts of this document.

Lastly, I would like to thank my parents and my sister, whose continual support has enabled every one of my accomplishments. Their advice, love, and encouragement have helped inspire and drive me to become who I am today. I would like to thank my incredible wife, who embarked with me on this adventure across the country, away from family and friends, to allow me to pursue this degree. She has endured with me through every failure and rejoiced in every triumph. Without her support, none of this would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AM . . . . . amplitude modulation

ARW . . . . . angle random walk

BCE . . . . . before the common era

CE . . . . . common era

DARPA . . . . . Defense Advanced Research Projects Agency

DCM . . . . . direction cosine matrix

deg/hour . . . . . degree(s) per hour

deg/sec . . . . . degree(s) per second

DR . . . . . dead reckoning

ECEF . . . . . earth-centered earth-fixed

ECI . . . . . earth-centered inertial

EKF . . . . . extended Kalman filter

ENU . . . . . east-north-up

EOSL . . . . . Electro-Optical Systems Laboratory

FBN . . . . . feature-based navigation

GLONASS . . . . . Russian Global Navigation Satellite System

GNSS . . . . . global navigation satellite system

GPS . . . . . global positioning system

GPU . . . . . graphics processing unit

GTRI . . . . . Georgia Tech Research Institute

Hz . . . . . Hertz

IMU . . . . . inertial measurement unit

INS . . . . . inertial navigation system

KF . . . . . Kalman filter

LGF . . . . . local geodetic frame

LiDAR . . . . . light detection and ranging

LORAN . . . . . long-range navigation

LU . . . . . lower upper

m . . . . . meter(s)

m/s . . . . . meter(s) per second

MEMS . . . . . micro-electromechanical systems

NED . . . . . north-east-down

PF . . . . . particle filter

PNT . . . . . precision navigation and timing

R&D . . . . . research and development

RMS . . . . . root-mean squared

SAR . . . . . synthetic aperture radar

SBAS . . . . . space-based augmentation system

SDE . . . . . stochastic differential equation

sec . . . . . second(s)

SLAM . . . . . simultaneous localization and mapping

SoOP . . . . . signals of opportunity

SPKF . . . . . sigma point Kalman filter

UAS . . . . . unmanned aerial system

UKF . . . . . unscented Kalman filter

UT . . . . . unscented transformation

VRW . . . . . velocity random walk

WGS84 . . . . . World Geodetic System 1984

WLAN . . . . . wireless local area network

SUMMARY

This thesis presents a modular navigation filtering framework specialized for use in a research and development environment. The developed framework emphasizes flexibility and modularity of filter components over computational performance in order to minimize the engineering work required to reconfigure the design for a new application. Modularization is accomplished by exploiting the natural mathematical interfaces of the Kalman Filter and related variants, while exercising the design discipline to truly maintain filter component separation. This approach results in a framework that retains compatibility with many Kalman Filter variants as well as different system models. The filter framework is qualitatively tested with simple system and sensor model implementations in two types of Monte Carlo simulation. The first set of tests utilizes trajectory data generated from a crude random walk model to check system stability and tuning. The second set of tests evaluates system performance using more realistic trajectory data generated with the X-Plane flight simulator. The test results demonstrate adequate performance of the simple models and overall viability of the design. Lastly, several improvements are proposed to increase the utility of the framework.

Chapter 1 contains a review of the relevant history to develop an understanding of the fundamental methods of navigation and how various technologies have applied these methods over time. Chapter 2 presents a mathematical description of the two fundamental methods of navigation and the relevant theory that enables fusion of the methods. This chapter concludes with a brief review of modern and future aiding sensor technologies. The design of the filter framework is provided in Chapter 3, which assumes a structured approach to achieve strict modularization of the framework components. These components include the filter mechanization, initial process model, inertial sensor models, and measurement models. This chapter discusses the selection and design of each component. The simulation framework utilized to test the filter is described in Chapter 4 along with the results

from two different types of flight simulation. The qualitative performance of the modular framework and initial model implementations is also analyzed. This thesis concludes in Chapter 5 with a summary of the research results and the identification of potential improvements and future directions.

CHAPTER 1
ORIGIN AND HISTORY

The navigation problem has a rich history of elegant and inventive solutions. From the innate human ability to recognize one's surroundings to the technology behind the modern global positioning system (GPS), all methods of navigation throughout history can be reduced to two fundamental techniques. The first and most intuitive technique is to observe distinct features in the environment and determine one's location, or localize, based on prior knowledge of the location of these features. This method is known as feature-based navigation (FBN).

The second method, known as dead reckoning (DR), involves predicting one's location by assuming that the desired effect of a control input was perfectly achieved. This method is embodied by the familiar example of counting one's steps while walking in a straight line. Given the number of steps, the assumed distance per step, and the direction of travel, overall distance from the starting point may be determined. These two fundamental methods are in some ways complementary. DR only provides location relative to a known starting point, and FBN is not always possible if the features are ambiguous or obscured. As complementary methods, both are essential to solving the navigation problem. This chapter discusses the evolution of this relationship over the course of human history.

In ancient times, mankind relied heavily on FBN due to its intuitive nature. As early as 600 BCE, advances in cartography provided knowledge of landmark locations to allow localization within a surveyed area [1]. Similarly, careful observations of the heavenly bodies led to the creation of celestial charts, enabling the sun and stars to serve as distant stationary landmarks. Around 300 BCE, the ancient Greeks exploited this information to precisely measure latitude. Longitude presented a more difficult problem. Due to Earth's rotation, celestial measurement of longitude requires knowledge of local time with respect to time at

the longitude datum (the Prime Meridian). Without the ability to precisely measure time, full global positioning based on the stars was not possible for many years. Similarly, DR was initially limited by a lack of precision in measuring time, compounded by the lack of education in simple arithmetic and geometry. Records exist of crude DR used by sailors in the Mediterranean Sea in the context of sailing towards a given star or against a seasonal wind for some coarse unit of time. The invention of the magnetic compass around 1100 CE removed uncertainty in at least one of the requisite parameters. Using the compass to determine bearing, DR proved useful in the relatively calm waters of the Mediterranean but suffered a great deal of error under the influence of currents in the open ocean.

The environment of the high seas provided one of the first significant challenges for navigational science. Without sight of land or accurate measurements of speed or time, little could be determined about true position, or more specifically, longitude [2]. Celestial methods were adapted for use at sea allowing vessels to sail horizontally along lines of latitude. However, without longitude, many lives were lost to the oceans. The severity of the problem drove the British to establish the Board of Longitude in 1714 which offered monetary prizes for scientific developments toward its solution. The resulting competition produced two distinct techniques. The first method estimated time at a remote port by measuring the current lunar distance to another celestial body and referencing the difference to predicted time in a nautical almanac. The second method tracked time at port using specially designed sea-worthy clocks, or chronometers, that were resistant to temperature, humidity, and corrosion in a marine environment.

At the time, the competition was marked by scandal and intense debate. In reality, the two competing methods shared a symbiotic relationship. The chronometer performed dead-reckoning in some sense by mechanically propagating time from a known starting point, accumulating error in the process. Through measuring external features, the lunar distance method provided a way to reset the chronometer and flush the accumulated error. However, the quality of lunar distance measurements was dependent on cloud cover

and other environmental factors. Chronometer measurements were (mostly) unaffected by these conditions. These properties emphasize an important relationship between the two fundamental methods of navigation that form a common thread through future developments. Despite the effectiveness of a hybrid technique, marine chronometers did not see widespread use until the early to mid 1800s. Manufacturing costs were reduced to the point that chronometers became affordable for most merchant seaman, with many government vessels carrying three for redundancy (a precursor to triplex redundancy in safety-critical systems).

The next significant advances in navigation technology came with developments in inertial sensing. Work in the 1800s led to the invention of the gyroscope [3]. In the early 1900s, gyroscopes were applied as a method to measure true heading. These so-called gyrocompasses replaced their magnetic counterparts that were prone to errors caused by local variations in the Earth's magnetic field. The other crucial inertial sensor, the accelerometer, resulted from research on strain gage technologies in the 1920s [4, 5]. When used in tandem, these inertial sensors enabled a new form of DR based on nearly direct measurement of the kinematic states of an object. This new capability improved on the crude existing methods of measuring these quantities while also eliminating the need to make observations of the remote environment. Similar to the chronometer, inertial sensors provided the ability to compute distance and orientation from some starting point with the promise of heightened measurement availability or less susceptibility to environmental conditions.

Inertial sensing technology matured tremendously over the years following World War II. The first inertial navigation system (INS) producing full position and orientation was developed in the 1950s. This achievement led to the integration, or fusion, of a variety of aiding sensors with inertial sensors to help decrease long-term drift due to the accumulation of errors in DR. However, methods of fusing data from the various sensors lacked optimality. During this time period, researchers from many different disciplines were converging on a solution to the sensor fusion and optimal estimation problems [6]. Published in 1960,

the Kalman filter (KF) presented the first widely-accepted solution. The KF optimally mechanized the generic fusion of FBN and DR techniques into an online state estimator. In reality, others had arrived at variants of the same solution at that time from a number of different approaches in statistics and linear algebra.

The KF was derived using a fairly rigid set of assumptions. Kalman and others immediately began work to extend the solution to accommodate diverse applications, most notably, the Apollo space program. This effort resulted in a large number of variations for different classes of problems. Important variants include the continuous time KF and the extended KF which handle continuous time and non-linear system dynamics, respectively. Other more exotic variants were also developed. The particle filter (PF), which was actually conceived before the KF, capitalized on statistics and the enhanced computing power available in the 1980s to estimate the full probability distribution of a particular navigation state.

While these fusion algorithms were under development, research continued toward improving inertial sensing technologies. One of the key improvements came with the rate gyroscope. Traditionally, gyroscopes measured rotation directly using a spinning mass suspended in a low-friction gimbal assembly. Rate gyroscopes introduced a way to instead measure rotation rate, removing the need for bulky gimbal assemblies. This simplification also created a new problem. INS designs up to this point placed the accelerometers on the gimbal platform. Through a number of techniques, the gimbal platform was slaved to a known bearing, allowing the measurement of accelerations in a stationary reference frame aligned to that bearing. New designs strapped the accelerometer directly to the body of the moving object. The problem was solved using a different set of motion equations and the method became known as strapdown navigation.

The invention of radio in the early 1900s generated a great deal of interest in its application to navigation. Radio was used for many years as a navigational aid prior to the major successes in inertial navigation [7]. By broadcasting signals from known locations,

radio navigation emulated traditional visual FBN, relying instead on invisible electromagnetic features. Numerous schemes were developed to allow human operators to determine their range and/or bearing [8]. One popular scheme in the 1930s utilized multiple transmitters separated by some distance that broadcast unique signals, sometimes in Morse code. These signals were selected to produce an clear tone through interference at the center of their overlapping region. Navigators could then find their deviation from a predetermined course by the audible interference of the two tones. Other systems provided range and bearing by outputting synchronized pulses from multiple distant transmitters [9]. The differences in times of arrival were used to define hyperbolic lines of position along the earth's surface. Given two such differences, navigators could determine their precise location as the intersection of two such lines on a navigational chart. This fundamental principle was employed by the long-range navigation (LORAN) system, operated widely around the world through a vast network of transmitters. These methods were later incorporated into INS technologies as a method to automatically reset accumulated DR error.

Despite its benefits, classical radio navigation was limited in coverage by land-based transmitter locations. Global navigation satellite system (GNSS), like the well-known GPS, solved this problem by moving these transmitters into orbit [10]. As artificial features in the sky, GNSS satellites were designed to transmit precise position and timing information. While GPS was under development in the U.S., other nations were also working on satellite navigation systems. Other operational GNSSs include the Russian Global Navigation Satellite System (GLONASS) and the European Galileo system. Modern survey-grade GNSS receivers employ space-based augmentation system (SBAS) or differential measurements using nearby base stations to achieve centimeter or even millimeter level positional accuracies.

Just as methods of celestial navigation were thwarted by a cloudy sky, GNSS was quickly found to suffer performance issues in areas with poor satellite visibility or high multipath interference. These deficiencies revived the need for alternative aiding technolo-

gies. As a result, new approaches to land-based radio navigation were developed in the last few decades that take advantage of wireless local area network (WLAN) or cellular networks [11]. Even more recently, research was conducted exploring navigation based on signals of opportunity (SoOP) like television or AM radio [12]. Other non-radio based solutions have been suggested that employ optical cameras for motion tracking or light detection and ranging (LiDAR) [13, 14, 15].

As one final historic note, developments in statistics, artificial intelligence, and robotics in the 1980s spawned a new discipline known as simultaneous localization and mapping (SLAM) [16]. SLAM incorporated real-time mapping into the navigation framework to allow for localization in unknown or time-varying environments. A number of computational solutions were devised, most of which utilize a variation of the KF or PF. The current challenge lies in producing effective and efficient implementations and system models for a given application.

This thesis seeks to describe navigation filtering and modern aiding sensor technologies from the perspective of the two fundamental methods of navigation. With this background developed, a modular and generic navigation filter architecture is suggested as a flexible testbed for R&D applications. The use case for the testbed is in lidar-aided inertial navigation. In this context, lidar aiding consists of online 3D environmental mapping and correlation with a priori feature maps, with clear connections to collaborative SLAM. Elements from the final system design will be used for internal research by the Electro-Optical Systems Laboratory (EOSL) at Georgia Tech Research Institute (GTRI).

CHAPTER 2
BACKGROUND THEORY

To begin a discussion on navigation theory, it is important to further define some terminology. The term navigation itself assumes a variety of definitions based on common usage. The consensus is that navigating refers to the act of finding a way to move from one location to another. For the sake of specificity, this notion is often differentiated within aerospace disciplines into the three separate tasks of guidance, navigation, and control. Navigation refers only to the process of determining one's location in the surrounding environment. Guidance and control then refer to planning a path from the current location to a desired location and successfully steering to maintain the planned path. In this context, the notion of location is often expanded to include other kinematic states of an object like velocity and orientation. Object state can be further generalized to include any number of parameters that represent the abstract state of an object in an abstract environment. This chapter will mathematically define FBN and DR, explore how these definitions fit within the KF framework, and discuss specific techniques and aiding sensors used in modern navigation.

2.1 Notation

This chapter will rely heavily on some of the more standard and simplistic notations in the field. Vector quantities are shown in bold ($\boldsymbol{u}$). Matrix quantities are uppercase and also in bold ($\boldsymbol{H}$). Variables corresponding to a true state of the system or a noise-free measurement are denoted as $\boldsymbol{x}$. Estimates of these quantities include a hat ($\hat{\boldsymbol{x}}$). A tilde indicates that the quantity represents a noisy measurement ($\widetilde{\boldsymbol{y}}$). Discrete-time variables use subscripts to denote the time index, as in $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}(kT)$. In this context, k denotes the sample index and T is some sampling interval.

2.2 Two Fundamental Methods

FBN, also known as position fixing in some of the literature, involves two steps. In the first step, the environment is observed. This step is sometimes referred to as the measurement process and is stated mathematically as Equation 2.1.

$$\widetilde{\boldsymbol{y}} = \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{v}) \tag{2.1}$$

In this equation, $\boldsymbol{h}$ is a function that generates an observation $\widetilde{\boldsymbol{y}}$ based on the current object state $\boldsymbol{x}$ and some observation noise $\boldsymbol{v}$ caused by imperfections in the measurement. For physical measurements, this function models the interaction between the physical environment and the sensor. As defined, the observation function $\boldsymbol{h}$ implicitly maintains an internal representation of features in the environment and produces observations accordingly. The observation $\widetilde{\boldsymbol{y}}$ may take any form and contain any number of features.

In the second step, localization is performed by relating the current observation to prior observations to determine the object location relative to the features. Oftentimes, the simplest way to perform this localization is by inverting the observation model.

$$\hat{\boldsymbol{x}} = \boldsymbol{h}^{-1}(\widetilde{\boldsymbol{y}}) \tag{2.2}$$

The inverted model in Equation 2.2 produces an estimate, $\hat{\boldsymbol{x}}$, of the true object state $\boldsymbol{x}$ from the measurement $\widetilde{\boldsymbol{y}}$. This simplistic approach exhibits several important shortcomings. Any noise in the original observation is completely ignored since its value is unknown, which will degrade the quality of the resulting estimate. This approach also assumes that no ambiguities exist in the mapping between observation and object state, resulting in the assumption that the function $\boldsymbol{h}$ is one-to-one and invertible. In other words, a valid observation of environmental features must exist at every possible object location but these observations may not be unique. In this case, observations still provide useful infor-

mation for localizing the object but must be handled in a less direct manner. This property is equivalent to the observability of the system.

In DR, a blind prediction of the future object state is made based on the previous object state and control input. This method simply predicts the result of a control input on a system over some period of time and assumes that the desired effect of the input was perfectly achieved. Equation 2.3 gives the mathematical form of the prediction process.

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \tag{2.3}$$

In Equation 2.3, the object state $\boldsymbol{x}_k$ is predicted by a system dynamics model $\boldsymbol{f}$ also known as the process model. The output of the process model is dependent on the previous object state $\boldsymbol{x}_{k-1}$, the previous control input $\boldsymbol{u}_{k-1}$, and some process noise $\boldsymbol{w}_{k-1}$ which represents imperfections in the model. It is important to note the introduction of the time index k indicating that this model represents the state transition of the object through time. This time dependence lies in contrast with the first method which required no explicit concept of time to localize the object. However, introducing time allows this method to circumvent the need for external observations which is a highly desirable property in certain applications.

The equation in Equation 2.3 is easily generalized to continuous time. In this case, inputs to the function $\boldsymbol{f}$ are converted to continuous-time functions and $\boldsymbol{f}$ produces the time derivative of the object state. As a finer point, no real system operating in continuous time is truly observation free because the passage of time must be accurately observed to predict the future system state. In that sense, this second method is a subset of the first for continuous-time systems. This method is distinguished from the first because direct measurement of time is not necessarily required for discrete-time systems. State transitions between iterations in discrete-time systems are often completely determined without direct measurement of time.

2.3 Sensor Fusion

Sensor fusion algorithms provide a way to optimally combine measurements from multiple sensors. In the context of modern navigation, sensor fusion techniques combine FBN methods with DR to generate an optimal state estimate in real time. A great number of techniques exist with many tuned for highly specific applications. This chapter will intuitively discuss some of the more general and common techniques applied to navigation.

2.3.1 The Linear Kalman Filter

The KF is the premier sensor fusion algorithm due to its relative efficiency and ease of use [6, 17, 18, 19]. The standard KF employs a two step prediction and correction process which corresponds closely to DR and FBN. The main difference is that the correction step does not perform direct localization, for example, by inverting the measurement function, as in pure FBN. Since this inverse does not often exist, the best state estimate for a given measurement is solved using least squares where the prediction equations act as constraints [20]. This approach is shown for a linear system as an illustration. The linear process and measurement equations are defined in Equation 2.4 and Equation 2.5.

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \approx \boldsymbol{F}_{k-1}\boldsymbol{x}_{k-1} + \boldsymbol{G}_{k-1}\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \tag{2.4}$$

$$\widetilde{\boldsymbol{y}}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k) \approx \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{v}_k \tag{2.5}$$

These models are simply the linear forms of those given in Equation 2.3 and Equation 2.1 with time indices added to the measurement model to indicate observations made at a specified time. The system is given in terms of the true state $\boldsymbol{x}$ rather than the state estimate $\hat{\boldsymbol{x}}$ since noise is involved. In general, the KF derivation assumes measurement and process noise inputs are zero mean and uncorrelated in time or with one another. Any deviation from these assumptions requires that correlations or biases are modeled as part of

the system state, an approach known as state augmentation [6]. The resulting least squares problem is given by Equation 2.6.

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$$

$$\begin{bmatrix} \widetilde{\boldsymbol{y}}_0 \\ \boldsymbol{G}_0\boldsymbol{u}_0 \\ \widetilde{\boldsymbol{y}}_1 \\ \boldsymbol{G}_1\boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{G}_{k-1}\boldsymbol{u}_{k-1} \\ \widetilde{\boldsymbol{y}}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_0 & 0 & 0 & \cdots & 0 \\ -\boldsymbol{F}_0 & \boldsymbol{I} & 0 & \cdots & 0 \\ 0 & \boldsymbol{H}_1 & 0 & \cdots & 0 \\ 0 & -\boldsymbol{F}_1 & \boldsymbol{I} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{H}_k \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_0 \\ \hat{\boldsymbol{x}}_1 \\ \vdots \\ \hat{\boldsymbol{x}}_{k-1} \\ \hat{\boldsymbol{x}}_k \end{bmatrix} \tag{2.6}$$

$$\hat{\boldsymbol{x}} = (\boldsymbol{A}^\mathsf{T}\boldsymbol{A})^{-1}\boldsymbol{A}^\mathsf{T}\widetilde{\boldsymbol{y}} \tag{2.7}$$

The solution to Equation 2.6 is computed using the standard left pseudo-inverse. The system of equations can be solved in this form by numerical software packages for small values of k, but the computation time will increase as more measurements are included. This increase is due to the fact that this solution includes estimates of past states conditioned on, or using information from, future measurements. Problems of this form are known as smoothing problems since all measurements corresponding to a certain window of time are considered in producing the optimal state trajectory over that same window. As demonstrated in this solution, the size of the problem grows with the time interval. For real-time estimation applications, a recursive or streaming solution is desired to increase online efficiency of the algorithm. This efficiency is often gained at the cost of optimality over the full trajectory.

The streaming solution to Equation 2.7 is found by factoring the quantity $\boldsymbol{A}^\mathsf{T}\boldsymbol{A}$ using block tridiagonal LU factorization. Such systems are easily solved since the tridiagonal

matrix can be factored into lower and upper triangular matrices. Using substitution, the problem is reduced to solving two triangular systems which only contain two nonzero terms in each row [21]. This improvement does not immediately result in a streaming solution. Recursion is added by exploiting the structure of the tridiagonal factorized solve algorithm to efficiently add new process and measurement equations in an incremental manner. As a result, the prediction and measurement update steps are handled by appending the row of the corresponding equation to the end of the matrix. In this form, the streaming solution produces the state estimate and a quantity known as the information matrix. The estimated state covariance is produced by inverting this information matrix using the matrix inversion lemma.

One of the great benefits of the KF is that it estimates its own uncertainty through the state estimate error covariance output allowing for health monitoring and other applications. As with all covariance matrices, the KF covariance estimate is symmetric and positive semi-definite. The resulting equations for the discrete-time linear KF are given in Equation 2.8 and Equation 2.9.

$$\begin{aligned} \hat{\boldsymbol{x}}_k^- &= \boldsymbol{F}_{k-1}\hat{\boldsymbol{x}}_{k-1}^+ + \boldsymbol{G}_{k-1}\boldsymbol{u}_{k-1} \\ \hat{\boldsymbol{P}}_k^- &= \boldsymbol{F}_{k-1}\hat{\boldsymbol{P}}_{k-1}^+\boldsymbol{F}_{k-1}^\mathsf{T} + \boldsymbol{Q}_{k-1} \end{aligned} \tag{2.8}$$

$$\begin{aligned} \boldsymbol{K}_k &= \hat{\boldsymbol{P}}_k^-\boldsymbol{H}_k^\mathsf{T}(\boldsymbol{H}_k\hat{\boldsymbol{P}}_k^-\boldsymbol{H}_k^\mathsf{T} + \boldsymbol{R}_k)^{-1} \\ \hat{\boldsymbol{x}}_k^+ &= \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\widetilde{\boldsymbol{y}}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_k^-) \\ \hat{\boldsymbol{P}}_k^+ &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\hat{\boldsymbol{P}}_k^-(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)^\mathsf{T} + \boldsymbol{K}_k\boldsymbol{R}_k\boldsymbol{K}_k^\mathsf{T} \\ &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\hat{\boldsymbol{P}}_k^- \end{aligned} \tag{2.9}$$

In these equations, the process and measurement noise covariances, $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$, are added to account for non-unit variances on the white noise inputs $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$. In effect, these covariances act as gains which are dependent on the statistics of the stochastic inputs, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$, but may be tuned in real-world applications. Equation 2.8 gives the so-called

time update or prediction equations corresponding to the DR step of the algorithm. These equations produce the *a priori* estimate $\hat{\boldsymbol{x}}_k^-$ and covariance $\hat{\boldsymbol{P}}_k^-$ which are denoted by the superscript minus. The *a priori* values incorporate the next state prediction determined through DR, but lack the most recent measurement. This measurement is incorporated during the FBN or correction step in the *a posteriori* estimate and covariance, $\hat{\boldsymbol{x}}_k^+$ and $\hat{\boldsymbol{P}}_k^+$, denoted by the superscript plus. The probabilistic definition of these quantities yields more insight. From the probabilistic standpoint, the KF estimates the expected values shown in Equation 2.10 and Equation 2.11.

$$\begin{aligned} \hat{\boldsymbol{x}}_k^- &= E[\boldsymbol{x}_k|\boldsymbol{y}_0, \cdots, \boldsymbol{y}_{k-1}] \\ \hat{\boldsymbol{P}}_k^- &= E[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^-)(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^-)^\mathsf{T}] \end{aligned} \tag{2.10}$$

$$\begin{aligned} \hat{\boldsymbol{x}}_k^+ &= E[\boldsymbol{x}_k|\boldsymbol{y}_0, \cdots, \boldsymbol{y}_k] \\ \hat{\boldsymbol{P}}_k^+ &= E[(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^+)(\boldsymbol{x}_k - \hat{\boldsymbol{x}}_k^+)^\mathsf{T}] \end{aligned} \tag{2.11}$$

The expected value after the prediction phase or *a priori* estimate, given in Equation 2.10, is conditioned on all but the most recent measurement $\widetilde{\boldsymbol{y}}_k$. From this perspective, it is clear that prediction is the same as simple DR. The *a posteriori* in Equation 2.11 includes the most recent measurement as expected. The full probabilistic derivation of the filter is developed by computing the expectations in Equation 2.10 and Equation 2.11 and inferring recursive equations from the results. The required gain $\boldsymbol{K}_k$ is derived from standard recursive least squares. This paper will not further discuss the probabilistic derivation of the linear KF, as this result should be sufficient until more advanced topics.

In many cases, the linear KF offers sufficient performance [6, 19]. For truly linear systems with uncorrelated white noise, the linear KF gives the optimal solution in the least-squares sense. Numerous methods exist to relax some of the linear KF requirements like uncorrelated zero-mean white stochastic inputs. Other methods exist which promote system stability, simplify online computation, or inform different representations of the system

dynamics. Simple extensions also exist for continuous-time systems. This paper will focus on methods that are widely used for navigation specifically.

2.3.2 The Extended Kalman Filter (EKF)

The most common KF variant is known as the extended Kalman filter (EKF). The innovation of the EKF is that it accommodates nonlinear process and measurement models through linearization of the models around the best current state estimate. Linearization is accomplished with a first-order Taylor series approximation using the derivatives of the model equations. Model linearization is shown in Equation 2.12 and Equation 2.13.

$$\begin{gathered} \boldsymbol{x_k} = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \boldsymbol{w}_{k-1}) \\ \boldsymbol{x_k} \approx \boldsymbol{f}(\hat{\boldsymbol{x}}^+_{k-1}, \boldsymbol{u}_{k-1}, 0) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}^+_{k-1}} (\boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}^+_{k-1}) + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w}}\right|_{\hat{\boldsymbol{x}}^+_{k-1}} \boldsymbol{w}_{k-1} \end{gathered} \tag{2.12}$$

$$\begin{gathered} \widetilde{\boldsymbol{y}}_k = \boldsymbol{h}(\boldsymbol{x}_k, \boldsymbol{v}_k) \\ \widetilde{\boldsymbol{y}}_k \approx \boldsymbol{h}(\hat{\boldsymbol{x}}^-_k, 0) + \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}^-_k} (\boldsymbol{x}_k - \hat{\boldsymbol{x}}^-_k) + \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{v}}\right|_{\hat{\boldsymbol{x}}^-_k} \boldsymbol{v}_k \end{gathered} \tag{2.13}$$

One subtle difference exists between these two equations. Rather than using the previous *a posteriori* state estimate $\hat{\boldsymbol{x}}^+_{k-1}$, the measurement update in Equation 2.13 is linearized about the current *a priori* estimate $\hat{\boldsymbol{x}}^-_{k-1}$ since it includes the most recent prediction and should be closer to the true value during correct operation. The prediction and correction equations for the EKF are given in Equation 2.14 and Equation 2.3.2, respectively.

$$
\begin{aligned}
\hat{\boldsymbol{x}}_k^- &= \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_{k-1}, 0) \\
\boldsymbol{F}_{k-1} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{k-1}, 0)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}^+} &\quad \boldsymbol{L}_{k-1} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}_{k-1}, 0)}{\partial \boldsymbol{w}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}^+} \\
\hat{\boldsymbol{P}}_k^- &= \boldsymbol{F}_{k-1}\hat{\boldsymbol{P}}_{k-1}^+\boldsymbol{F}_{k-1}^\mathsf{T} + \boldsymbol{L}_{k-1}\boldsymbol{Q}_{k-1}\boldsymbol{L}_{k-1}^\mathsf{T}
\end{aligned}
\tag{2.14}
$$

$$
\begin{aligned}
\boldsymbol{H}_k = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}, 0)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k^-} &\quad \boldsymbol{M}_k = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}, 0)}{\partial \boldsymbol{v}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k^-} \\
\boldsymbol{K}_k &= \hat{\boldsymbol{P}}_k^-\boldsymbol{H}_k^\mathsf{T}(\boldsymbol{H}_k\hat{\boldsymbol{P}}_k^-\boldsymbol{H}_k^\mathsf{T} + \boldsymbol{M}_k\boldsymbol{R}_k\boldsymbol{M}_k^\mathsf{T})^{-1} \\
\hat{\boldsymbol{x}}_k^+ &= \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k[\widetilde{\boldsymbol{y}}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_k^-, 0)] \\
\hat{\boldsymbol{P}}_k^+ &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\hat{\boldsymbol{P}}_k^-(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)^\mathsf{T} + \boldsymbol{K}_k\boldsymbol{M}_k\boldsymbol{R}_k\boldsymbol{M}_k^\mathsf{T}\boldsymbol{K}_k^\mathsf{T} \\
&= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\hat{\boldsymbol{P}}_k^-
\end{aligned}
\tag{2.15}
$$

These equations are strikingly similar to those in the linear KF except for two key differences. The first difference is that the state and measurement predictions are evaluated using nonlinear instead of linear model functions. The second difference is that the process and measurement noise covariances are linearly transformed by their respective sensitivity matrices $\boldsymbol{L}_{k-1}$ and $\boldsymbol{M}_k$. In this case, the model state sensitivities $\boldsymbol{F}_{k-1}$ and $\boldsymbol{H}_k$ are used to linearly propagate the covariance as in the linear KF. These operations are the direct result of the first-order Taylor series approximation making the overall estimate of the state distribution only accurate to the first order.

The EKF has been proven highly successful and is the most commonly used filter for navigation [10]. For many applications, the first-order approximation is sufficient. For highly nonlinear systems, indirect approaches exist which simplify nonlinearities by estimating error states rather than the system states themselves [22]. Higher order approxima-

tions can be employed if this sort of transformation is not adequate or if the state distribution may become multimodal due to ambiguous sensor measurements.

2.3.3 Continuous/Discrete-Time Kalman Filtering

One of the most appealing properties of the KF framework is its generality and resulting flexibility. For most real-world problems, it is convenient to describe systems through continuous-time models, but digital computers and sensors operate in discrete time. The prediction/correction framework simplifies development of a hybrid continuous-discrete time KF. In this case, the system dynamics handled in the prediction step are processed as a continuous-time system. Measurement updates are then processed as discrete-time events occurring when new sensor data are received. As a result, this technique easily accommodates multi-rate sensor data. One key distinction in the continuous-time KF is that the process model equation actually describes the first time derivative of the state, $\dot{\boldsymbol{x}}$, rather than the next discrete state, $\boldsymbol{x}_k$. The process model prediction, $\dot{\hat{\boldsymbol{x}}}$, must be integrated to produce the *a priori* state estimate, $\hat{\boldsymbol{x}}_k^-$, requiring the choice of a suitable numerical integration method to solve the stochastic differential equation (SDE). Some familiar schemes are the Euler-Maruyama and modified Heun methods [23, 24]. These methods produce first and second order approximations of the integral, respectively, and are strikingly similar to the forward Euler and explicit trapezoidal methods for deterministic differential equations. However, both of these methods properly account for scaling of the discretized noise. Higher order schemes, like SDE extensions to Runge-Kutta methods, can also be employed at the cost of additional computation. Discretization will be further discussed in Section 3.3.4.

2.3.4 The Particle Filter (PF)

One perspective of the KF is that it estimates the unimodal probability distribution of a random variable by estimating its mean and covariance as they are transformed by the sys-

tem over time. In a more general sense, it is possible to estimate any arbitrary transformed distribution of a random variable using a discrete approximation. This process is known as Bayesian state estimation and is implemented recursively by the PF. To perform this operation, some number of samples of the initial distribution, or particles, are transformed by the system dynamics model with representative noise injected [25]. This step generates *a priori* particles, $\hat{\boldsymbol{x}}_{k,i}^{-}$, which represent a set of possible *a priori* state estimates. Noise for each particle, $\boldsymbol{w}_{k-1,i}$, is generated by sampling an appropriate distribution. This step effectively performs prediction, or DR, through Monte Carlo simulation.

To apply a measurement update, particles are weighted based on their likelihood given the most recent measurement. Likelihood is determined by computing the conditional probability distribution $p(\widetilde{\boldsymbol{y}}_k|\hat{\boldsymbol{x}}_{k,i}^{-})$ for each particle. For the case of additive Gaussian measurement noise, this distribution can be computed directly using the measurement residual $\widetilde{\boldsymbol{y}}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k,i}^{-}, 0)$ and the measurement error covariance, $\boldsymbol{R}_k$. A new set of particles are then generated based on these likelihoods, representing the *a posteriori* distribution. This resampling of the particles can be performed in a variety of ways, and the method of resampling often dictates the overall performance of the PF. The resampled particles, denoted as $\hat{\boldsymbol{x}}_{k,i}^{+}$, form a discrete approximation of the resulting distribution, allowing for the computation of mean and covariance.

The PF offers several substantial benefits at the potentially high computational expense of evaluating the system models and probabilities for every particle. One such benefit is that the PF requires no specific knowledge of the process or measurement dynamics besides the ability to evaluate each for a given set of inputs. In this sense, the PF is a "black box" filter as it can be easily mechanized in an abstract and modular manner without requiring model derivatives. Another benefit is that the PF enables approximation of arbitrary or multimodal distributions. Given a sufficient number of particles, all modes in the distribution are simultaneously propagated. In the context of navigation, the effect of this capability is that the filter actively tracks multiple hypotheses for a state estimate. The tradeoff, in this

case, comes with the expense of tracking a sufficient number of particles to represent the underlying distribution.

Even if sufficient particles exist, performance of the PF can suffer due to sample impoverishment. This condition occurs when the particle distribution loses statistical diversity during resampling, causing a decrease in the effective number of particles. A variety of methods exist to alleviate this condition, some of which rely on adding artificial noise to move each particle into a more relevant region of the state space. Hybrid methods also exist which combine the PF and simpler KF variants. The extended Kalman particle filter employs an EKF for performing prediction and correction of each particle. After correction, particle likelihoods are computed and resampling occurs as in the regular PF. Another common use of the hybrid approach, known as Rao-Blackwellization, reduces the overall dimensionality and computational burden of the problem. In this approach, another form of the KF is used to estimate states that are linear or unimodal. The PF then provides estimates for the subset of states with nonlinear dynamics or multimodal distributions.

2.3.5 The Unscented Kalman Filter (UKF)

The great disadvantage of the PF derived from the computational expense of tracking the large number of particles required to accurately represent a state distribution. This additional computational burden is unnecessary for problems that are not necessarily multimodal but still highly nonlinear. Sigma point Kalman filter (SPKF) solve this problem by performing a similar operation over a small number of deterministically sampled particles, known as sigma points in this context [26]. These sigma points are sampled from the original distribution such that they completely capture all the information required to approximate the distribution to the second order or higher. The sigma points are then passed through the nonlinear process model to generate an approximation of the transformed distribution, from which the mean and covariance are computed.

Several methods of sigma point selection exist. The most common method is known as the unscented transformation (UT) and serves as the basis for the unscented Kalman filter (UKF) [27, 28]. The UT is shown in Equation 2.16 and Equation 2.17 for a system of N states with mean $\hat{\boldsymbol{x}}$, covariance $\boldsymbol{P}$, and a nonlinear transformation $\boldsymbol{h}$. In this context, i denotes the sigma point index defined on the interval $1 \leq i \leq 2N$, and $(\cdots)_i$ is an operator retrieving the ith row of the matrix argument.

$$\hat{\boldsymbol{x}}^{(i)} = \hat{\boldsymbol{x}} + \begin{cases} (N\boldsymbol{P})_i^{\mathsf{T}} & 1 \leq i \leq N \\ -(N\boldsymbol{P})_i^{\mathsf{T}} & N+1 \leq i \leq 2N \end{cases} \tag{2.16}$$

$$\hat{\boldsymbol{y}}^{(i)} = \boldsymbol{h}(\hat{\boldsymbol{x}}^{(i)}) \tag{2.17}$$

Nonlinear transformation of the sigma points occurs in Equation 2.17. The orthogonal eigendecomposition of Equation 2.16 reveals that the sigma points are perturbations about the mean value along principal axes defined by the covariance $\boldsymbol{P}$. The perturbed distance from the mean is scaled by the product of the total number of states and the standard deviation in the respective principal direction. Therefore, the sigma points lie on a hyperellipsoid centered about the mean. However, other interpretations of the matrix square root are equally valid and produce similar results [27]. This formulation of the UT utilizes $2N$ sigma points and makes the implicit assumption that the original distribution is symmetric [29]. The original UT, as presented in [27], makes no such assumption. In its original form, the UT requires $2N+1$ sigma points, with the mean serving as the extra point, and employs an additional scaling parameter to shape sigma point placement. Although more general, this method was omitted from the present research until required to improve performance.

With this definition of the UT, it is possible to describe the operation of the UKF. This derivation of the UKF deviates from previous system model definitions in that the stochastic inputs, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$, must be additive as in Equation 2.18 and Equation 2.19. However, it is possible to adapt this definition to non-additive noise using state augmentation.

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{w}_{k-1} \tag{2.18}$$

$$\widetilde{\boldsymbol{y}}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{2.19}$$

During the prediction step, the UKF selects sigma points as before based on the *a priori* state covariance estimate, $\hat{\boldsymbol{P}}_{k-1}$. Prediction proceeds in a straightforward manner using the process model given in Equation 2.18. This process is shown in Equation 2.20 and Equation 2.21.

$$\hat{\boldsymbol{x}}_{k-1}^{(i)} = \hat{\boldsymbol{x}}_{k-1}^{+} + \begin{cases} (N\hat{\boldsymbol{P}}_{k-1}^{+})_i^{\mathsf{T}} & 1 \leq i \leq N \\ -(N\hat{\boldsymbol{P}}_{k-1}^{+})_i^{\mathsf{T}} & N < i \leq 2N \end{cases} \tag{2.20}$$

$$\hat{\boldsymbol{x}}_k^{(i)-} = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}^{(i)}, \boldsymbol{u}_{k-1}) \tag{2.21}$$

$$\hat{\boldsymbol{x}}_{k,uc}^{-} = \frac{1}{2N} \sum_{i=1}^{2N} \hat{\boldsymbol{x}}_k^{(i)-} \tag{2.22}$$

$$\hat{\boldsymbol{P}}_k^{-} = \boldsymbol{Q}_{k-1} + \frac{1}{2N} \sum_{i=1}^{2N} (\hat{\boldsymbol{x}}_k^{(i)-} - \hat{\boldsymbol{x}}_{k,uc}^{-})(\hat{\boldsymbol{x}}_k^{(i)-} - \hat{\boldsymbol{x}}_{k,uc}^{-})^{\mathsf{T}} \tag{2.23}$$

The *a priori* state and covariance estimates are then computed directly from the sigma points using the standard formulas in Equation 2.22 and Equation 2.23. The process covariance, $\boldsymbol{Q}_{k-1}$, is included to add the effect of model uncertainties to the estimate. The correction step, in Equation 2.24, computes a new set of sigma points representing the distribution of the *a priori* estimate. These new sigma points are passed through the measurement model to produce an approximation of the predicted measurement distribution.

The measurement mean and covariance are then calculated from this distribution. These steps are shown in Equation 2.25, Equation 2.26, and Equation 2.27.

$$\hat{\boldsymbol{x}}_k^{(i)} = \hat{\boldsymbol{x}}_k^- + \begin{cases} (N\hat{\boldsymbol{P}}_k^-)_i^{\mathsf{T}} & 1 \leq i \leq N \\ -(N\hat{\boldsymbol{P}}_k^-)_i^{\mathsf{T}} & N < i \leq 2N \end{cases} \tag{2.24}$$

$$\hat{\boldsymbol{y}}_k^{(i)} = \boldsymbol{h}(\hat{\boldsymbol{x}}_k^{(i)}) \tag{2.25}$$

$$\hat{\boldsymbol{y}}_{k,uc} = \frac{1}{2N}\sum_{i=1}^{2N} \hat{\boldsymbol{y}}_k^{(i)} \tag{2.26}$$

$$\boldsymbol{P}_y = \boldsymbol{R}_k + \frac{1}{2N}\sum_{i=1}^{2N} (\hat{\boldsymbol{y}}_k^{(i)} - \hat{\boldsymbol{y}}_{k,uc})(\hat{\boldsymbol{y}}_k^{(i)} - \hat{\boldsymbol{y}}_{k,uc})^{\mathsf{T}} \tag{2.27}$$

$$\boldsymbol{P}_{xy} = \frac{1}{2N}\sum_{i=1}^{2N} (\hat{\boldsymbol{x}}_k^{(i)} - \hat{\boldsymbol{x}}_k^-)(\hat{\boldsymbol{y}}_k^{(i)} - \hat{\boldsymbol{y}}_{k,uc})^{\mathsf{T}} \tag{2.28}$$

The final statistic needed to complete the measurement update is the cross-correlation matrix relating predicted state estimates to predicted measurements, shown in Equation 2.28. The measurement update is completed by computing the gain, $\boldsymbol{K}_k$, and the *a posteriori* estimates as in Equation 2.29, Equation 2.30, and Equation 2.31.

$$\boldsymbol{K}_k = \boldsymbol{P}_{xy}\boldsymbol{P}_y^{-1} \tag{2.29}$$

$$\hat{\boldsymbol{x}}_{k,uc}^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\widetilde{\boldsymbol{y}}_k - \hat{\boldsymbol{y}}_k) \tag{2.30}$$

$$\hat{\boldsymbol{P}}_k^+ = \hat{\boldsymbol{P}}_k^- - \boldsymbol{K}_k\boldsymbol{P}_y\boldsymbol{K}_k^{\mathsf{T}} \tag{2.31}$$

Other forms of the SPKF differ from the UKF presented here only in the weighting of each sigma point. These weights precede each sigma point in the mean and covariance calculations, leading to nearly identical formulas for the generic SPKF.

The UKF has several appealing properties when compared to the EKF and PF. The UKF does not perform any linearization of the process and measurement models avoiding the need to compute derivatives. The UKF achieves a considerable performance improvement over the EKF at little additional computational expense. Like the PF, the UKF is also a "black box" filter but requires far less processing to accurately estimate a unimodal distribution. However, the UKF still only produces an approximation of a unimodal distribution and suffers performance issues with multimodal distributions. As with the other variants, the UKF has a number of useful extensions for handling special cases like non-additive or correlated noise [6, 30].

It is important to note that all of these filtering methodologies require a fairly detailed understanding of the system noise characteristics. Therefore, accurate system modeling is critical to achieve satisfactory performance from any of these sensor fusion algorithms. The next section will survey the multitude of modern sensing techniques employed in navigation to develop such an understanding, with specific emphasis on inertial sensors and strapdown architectures.

2.4 Modern Techniques and Aiding Sensors

Modern navigation employs a large variety of aiding sensors and techniques. Most of these methods rely on an inertial measurement unit (IMU) for acceleration and angular rate measurements to allow DR. The IMU is often rigidly fixed to the tracked object in a configuration known as strapdown navigation. Strapdown systems present some unique challenges compared to more classical techniques. In this configuration, measurements in the respective inertial sensor reference frames must be transformed into the body frame of the object and then again into the desired global frame. Each transformation potentially

includes a three-dimensional translation and rotation which can be represented in several different ways. Usually, translations take the form of a simple 3D vector. Rotations can be represented by direction cosine matrices (DCMs), Euler angles, and quaternions, among other representations.

Transforming body-fixed accelerations into some global frame requires knowledge of the object's attitude. This requirement creates an interdependence between two otherwise independent measurements, causing a correlation in the measurements and allowing uncertainties in attitude to affect velocity and position. In terrestrial applications, angular rate measurements detect the rotation of the earth projected into the body frame, which must be taken into account depending on the defined global reference frame. Coupling in the system dynamics due to fixing the sensors to the body frame exacerbates the already inherent drift caused by sensor biases. Despite these additional challenges, strapdown navigation remains the preferred method due to the corresponding weight and cost savings. As a result, aiding sensors like GPS are required to remove accumulated drift. However, unlike the sensor fusion algorithms previously discussed, aiding sensor models cannot be generalized at such a high level. This section discusses different methods used to correct inertial sensors in order to build an understanding of requirements for a generalized and modular navigation algorithm. Aiding sensor technologies can be categorized by whether man-made or natural features are observed. Man-made features are generally beacons which actively provide or influence some natural electromagnetic signal to allow localization. These features are usually designed specifically for navigation purposes with a few recent exceptions. Systems using natural features simply observe an ambient electromagnetic signal or field to measure motion. Natural features include the stars or the earth's gravitational field. These categories provide some rough high-level division between methods but there is considerable overlap in practice.

2.4.1 Standard GNSS-Aided Inertial Navigation

The most commonly used method of navigation is known as GNSS-aided inertial navigation [31]. GNSS constellations in orbit, like the familiar GPS, transmit precise timing and satellite orbital state information to passive receivers using coded signals [32]. Using the encoded information, each receiver is able to estimate the time of flight of the signal. This estimate, called a pseudorange, includes the true time of flight and an error corresponding to the offset between the receiver and GNSS time base. In the ideal case, only three ranges to known satellite locations are required to geometrically determine the receiver location through trilateration [10]. Each range defines the edge of a sphere centered around a satellite on which the receiver lies. The intersection of the edges of three spheres is at most two points, one of which is far from the surface of the earth and represents an invalid solution. The clock offset present in each pseudorange introduces an additional variable which prevents the perfect intersection of the spheres on their edges. Fortunately, the clock offset is common among all pseudoranges measured by a given receiver. The ambiguity added by this offset is resolved by using a pseudorange measurement from a fourth satellite. The clock offset is estimated as the value that causes the edges of all four spheres to intersect at a single point. The resulting range measurements allow calculation of position. Velocity is determined through differentiation of the position estimates or, more accurately, by using the Doppler shift of the signal. This description only provides an overview of high level operation as interpreting GNSS signals requires a fairly complicated process of signal correlation and decoding, beyond the scope of this thesis.

GNSS-aided INSs are grouped by the level of integration between the GNSS receiver and the navigation filter (presumably, some sort of KF). Tightly-coupled architectures generally include the clock offset and perhaps other internal GNSS states in the navigation filter. The filter is then responsible for the computations usually performed in the receiver. Such systems offer high accuracy but relatively low modularity as the central filter becomes, in some sense, monolithic. These architectures are common among survey-grade

INSs [33]. Loosely-coupled architectures incorporate the position and velocity outputs of the receiver as measurement updates for the navigation filter but rely on the receiver to calculate these quantities from pseudoranges. Loosely-coupled architectures are naturally more modular and simpler to implement but do not optimally combine all of the available information. Therefore, the theoretical performance of loosely-coupled architectures is inferior to that of tightly-coupled systems.

As an aiding sensor, GNSS is notable for its global coverage and high accuracy. Using a tightly-coupled architecture and satellite or land-based correction services, position accuracies on the centimeter scale are achievable. However, there are many conditions that can hinder or completely disable GNSS operation. Signals are susceptible to different kinds of interference over the relatively long distance between satellites and receivers. Poor satellite coverage in urban or heavily forested areas and poor atmospheric conditions may cause multipath interference or additional propagation delays. Indoor, subterranean, or submarine environments may block GNSS signals entirely. Other sources of interference are malicious in nature as GNSS jamming and spoofing have become inexpensive to achieve. These shortcomings have led to considerable research for alternatives to GNSS.

2.4.2 Non-GNSS Beacon-Aided Systems

Non-GNSS beacon-aided systems utilize man-made features that emit or modify ambient electromagnetic signals or fields. These systems vary in practicality depending on whether their application is commercial or military, and, like GNSS, remain sensitive to jamming and other interference due to their use of the electromagnetic spectrum. One proposed concept employs rapidly-deployable unmanned aerial system (UAS) swarms or low-altitude satellites to provide local positioning on a regional scale [11]. These systems act much like GNSS but are considered more immune to jamming due to the design reconfigurability afforded by rapid deployment. Unlike GNSS, two-way ranging schemes exist which improve accuracy at the cost of increased receiver observability in military applications.

Another proposed technique, known as peer-to-peer positioning, uses a collaborative scheme to improve the positioning accuracy of a group of receivers [34, 35]. In this case, each receiver employs inertial sensors to estimate its own position using DR. Each receiver then broadcasts a signal via radio frequency or ultrasonic transceivers which peers may use to determine relative ranges. In some cases, a subset of receivers may have access to GNSS, providing GNSS-denied peers within range the ability to localize in a global coordinate frame. This technique is a popular topic for pedestrian navigation and indoor positioning systems. Similar to the previous technique, this method requires active transmissions for ranging which is unappealing for many military applications.

The final method in this category exploits SoOP. SoOP include any available signal, such as radio, television, telecommunications, cellular, and Wi-Fi networks. One of the key difficulties of this method is that the precise time, and often the origin, of signal transmission is generally unknown. To accommodate this deficiency, SoOP methods use SLAM to generate a map of detected signals and their corresponding intensities at different locations [36]. This map of so-called signal fingerprints provides a source for measurement updates to aid in DR with an IMU. Given ambiguities in the fingerprint map, a PF is often required to maintain different hypotheses for the current position. In urban areas with a multitude of available signals, SoOP methods have demonstrated accuracies that rival and even surpass GNSS, especially in high multipath environments.

2.4.3 Imaging-Based Techniques

Another category of aiding technology extracts the kinematic state information of an observer from two or three-dimensional images of the environment. This method is distinct from beacon-aided methods in that the full image provides this information rather than any specific point source. Images may capture visible light or any other band of the spectrum. Some common sensor technologies in this category include monocular cameras, LiDAR, doppler, and synthetic aperture radar (SAR) [12, 13, 37]. Data collected from these sensors

may be correlated in real time with an existing map or database to produce state estimates. This method is employed in standard terrain-reference and lidar-aided navigation [38, 22]. Images from a single sensor may also be correlated during motion to estimate the change of state. Simple usage in this fashion provides measurements of attitude or position changes with more complex usage likely falling under the category of SLAM.

All imaging-based techniques often apply a similar approach. An algorithm automatically searches for unique features in the image according to certain suitability criteria. For visible light images, algorithms search for distinct features like corners. The remaining task is to correlate the detected features with those found in an existing image to determine the transformation between images. This task can be accomplished by comparing discrete features or whole regions of the image through correspondence matching, in which case, feature detection is not required as a separate step [39]. These techniques have demonstrated significant ability in stabilizing drifting IMU-based systems. However, correlation requires unambiguous, feature-rich images to avoid a poor match. These methods are also hindered by degraded visual environments for the required wavelengths, like those caused by fog or dust storms.

2.4.4 Pure Inertial Navigation

The final notable method does not necessarily fit this discussion of aiding sensor technologies. The last technique to aid inertial navigation is to simply improve IMU technology. From a simplistic perspective, the IMU itself is the ideal sensor for navigation. Given an accurate starting location and no sensor error, the IMU would be capable of perfectly tracking an object only using observations of the internal condition of its accelerometers and gyroscopes. Excluding electromagnetic interference, no method exists to allow an outside party to jam or manipulate these sensor readings, and no weather conditions can obscure or otherwise influence these measurements. Aside from sensor errors, this simple analysis neglects another important detail: accelerometers do not actually measure true

acceleration, but rather, specific force. The acceleration of gravity is not detected by the accelerometer, making it impossible to use for navigation during free-fall, which is common in orbital or interplanetary applications. This issue is remedied by adding gravity back into the acceleration using an accurate gravity model. In reality, accelerometers and gyroscopes exhibit a variety of errors and biases, and board-level clocks used for DR are relatively imprecise. Therefore, a significant amount of modern research is aimed at improving IMU performance in these areas.

Modern research in this area falls under the category of micro precision navigation and timing (PNT) [40]. This research focuses on chip-scale atomic clocks and micro-electromechanical systems (MEMS) for timing and inertial sensing due to their reduced cost, size, weight, and power [41]. Some specific research thrusts within this field include mitigating the temperature sensitivity and calibration drift typical of current MEMS inertial sensors. Cold-atom microsystems are one solution to some of these difficulties which use atomic wavelength lasers to control the temperature of clock and inertial sensing components at an atomic scale [3]. Other research thrusts seek to develop miniaturized rate-integrating gyroscopes that measure angular displacement rather than rate to avoid accumulating attitude error during DR.

Another interesting research thrust in this area involves using arrays of inexpensive inertial sensors to decrease long-term drift [42, 43]. In one such approach, sensors are selectively grouped during manufacturing through constrained optimization of their combined drift rate. This process results in a much lower combined bias for the group than for any individual sensor and results in a lower combined drift rate than afforded by simple averaging of the measurements [44]. This approach decreases IMU cost but requires more computational power and more complicated algorithms to fuse the measurements [45, 46]. Unfortunately, the method does not scale indefinitely as the inclusion of more sensors increases bias correlations caused by temperature changes [47]. Since the primary focus of this method is low cost, performance tends to fall far short of that required for strategic or

survey applications. Overall, pure inertial navigation will remain an important topic due to its appealing properties. The recent infusion of funding by organizations like the Defense Advanced Research Projects Agency (DARPA) will promote advances to this field in the near future and continue to move navigation towards more reliable and less vulnerable methods.

CHAPTER 3
TECHNICAL APPROACH

The primary purpose of this work is to develop a generic and modular navigation filter architecture suitable for use in a research and development environment. The resulting architecture should maximize compatibility to many different types of aiding sensors, with a specific emphasis on GNSS and LiDAR drawn from internal research goals within EOSL. This section provides a comprehensive definition of the selected architecture and initial process models while ensuring a high degree of flexibility for future changes to the system. To further define the design approach, several qualitative properties are loosely imposed. The filter design must seek to accomplish the following goals:

1. Achieve satisfactory performance for reasonably nonlinear process and measurement models typical of most terrestrial environments.
2. Ensure applicability to a variety of operating environments and vehicles.
3. Minimize the scope of work required to introduce changes to the process and measurement models.
4. Modularize components where possible to avoid dependencies across the design.
5. Generalize interfaces between aiding sensors and the filter to isolate sensor-specific details.
6. Maintain compatibility with common SLAM extensions.

These goals are purely qualitative and are derived from a rough estimate of future requirements. As with most engineering problems, a single solution is rarely optimal for a broad range of cases and generalization comes with a cost. This design allows for an increase in computational complexity to achieve these goals. To alleviate any excess computation, a highly-parallel computing architecture is assumed, for example, using graphics process-

ing unit (GPU) acceleration. Assuming cost is not prohibitive, it is apparent that a design with these properties is also suitable for any production application with a large number of managed configurations, such as a platform with a variety of available sensor suites that could optionally serve as aiding sensors to a central navigation system. With these applications in mind, the design proceeds by defining the overall system architecture and then the individual filter components, with brief review of common methodologies where necessary.

3.1 Modularized Architecture

Navigation system architectures range from tightly to loosely-coupled systems [10]. Tightly-coupled systems require detailed process and/or measurement models and fuse measurements in a centralized filter. These systems potentially offer the best performance since correlations between measurement errors are highly visible within the centralized filter. The trade-off for this high level of accuracy is a decrease in modularity and flexibility. Loosely-coupled systems compromise this performance to achieve better modularity by shifting processing responsibilities back to the aiding sensors. In this case, sensors produce more refined measurements, often through internal filtering, which are used in their processed form by the centralized navigation filter. In a federated system, filtering is further delegated to each individual sensor and the centralized filter is reduced to performing pure data fusion, often accomplished through standard least squares. Federated systems employ a cascaded filter configuration and must be carefully managed since filtered outputs often contain time correlation which violates one of the key assumptions for inputs of most Kalman filtering frameworks. The design in this research attempts to modularize a centralized filter through software architecture rather than federated filtering.

The standard KF framework inherently provides much of the structure required for a generalized design. The first step in the design is to reorganize the generalized components of this framework to achieve greater modularity. The revised structure serves as a guide to software implementation. A block diagram of the typical prediction and measurement

processes is shown in Figure 3.1. This general architecture is afforded by the KF framework and applies to all KF variants discussed to this point.

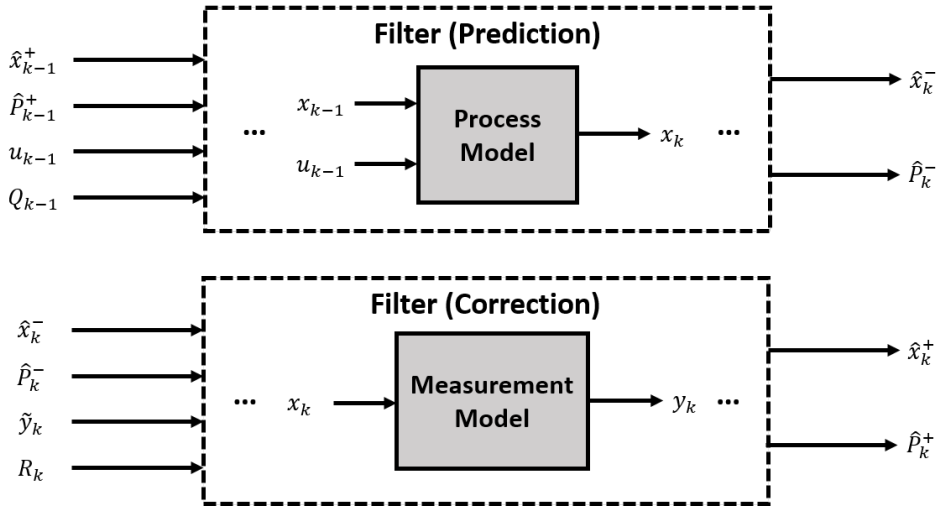

Figure 3.1: Standard Kalman Filter Framework Architecture

As shown in Figure 3.1, all KF variants require some internal notion of the process and measurement models. The primary insight for modularization is to remove the model implementations from their specific usage inside the filter, allowing generalization of the filter mechanization. Abstraction in this manner is possible due to the generalized inputs and outputs, or interface, of the model functions. This design was certainly the original intent for the mathematical KF framework and produces clear benefits from a software design standpoint. The degree and effectiveness of this abstraction depends on the specific type of filter. The process and measurement model implementations may take different forms based on the filter requirements. For the simplest case, this implementation takes the form of a the standard prediction and measurement processes following the common interfaces defined mathematically in Equation 2.1 and Equation 2.3. This case is depicted exactly by Figure 3.1 and is preferred since the model functions must already be defined for the problem setup. More complicated cases require a combination of the model function and

additional information (not depicted in Figure 3.1) like the model derivatives, as required by the EKF. The general architecture allows for the process model implementation to be run arbitrarily with any inputs depending on the needs of the filter. The PF and UKF use this capability to run the process and measurement models on each particle or sigma point.

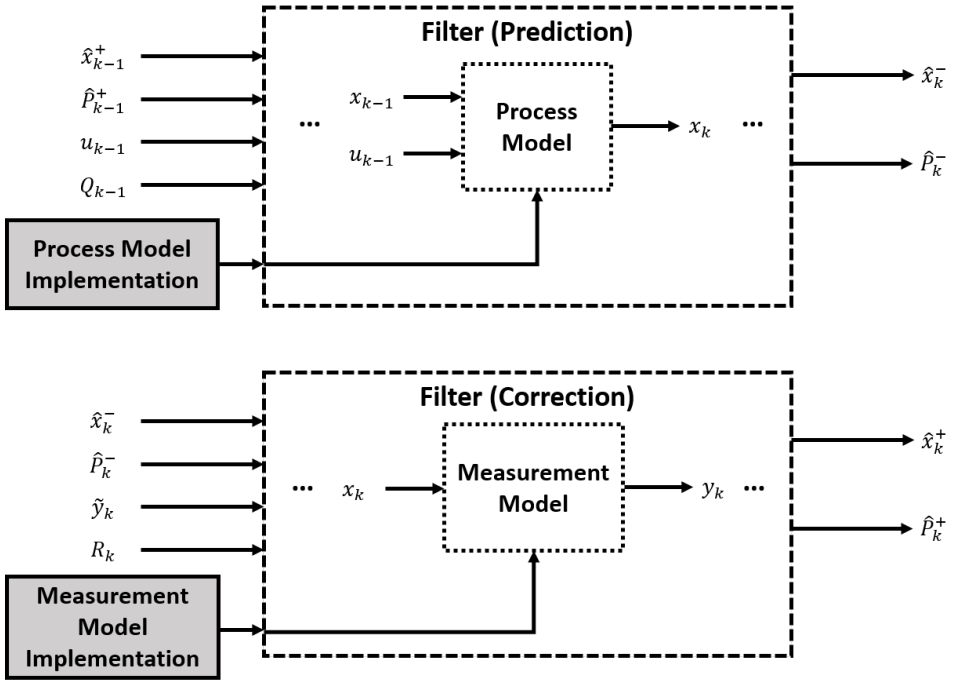

Figure 3.2: Modularized Kalman Filter Framework Architecture

The modularized abstraction is developed in Figure 3.2. This architecture segregates information and minimizes the knowledge of application details required by the central filter. These aims are accomplished through common mathematical interfaces, like those defined for the model functions, which map directly to software interfaces. The filter implementation in software is then completely agnostic to which version of the model function is actually invoked. As a result, the model functions become another input to the filter. Modularization also has the effect of minimizing design dependencies, and thus, reducing the work required to reconfigure the filter for a new application. The primary drawback, in this

case, is the loss of visibility into specific implementations of each functional component. Therefore, a careful balance is required to achieve the maximum utility from a modular design without obscuring critical relationships between components.

As a side note on implementation, many programming languages support abstraction of the process and measurement models through function handles or pointers, anonymous functions, or polymorphism. The simplest implementation using these mechanisms provides a small layer of modularization, but still requires the models to exist within the same code base as the filter itself. Adding another layer of separation, the models can be implemented in a dynamically-loaded software library, similar to a typical hardware driver. Dynamic loading prevents the centralized filter software from requiring any built-in information about the model functions, besides some sort of runtime configuration to indicate which models should be loaded. Special performance considerations must be taken into account when designing deterministic software with dynamic loading, which are beyond the scope of this present research. One last method to further separate the model implementations is inspired by the federated architecture. This method relies on the sensor hardware to run the sensor models, completely isolating the centralized filter from these details. However, this solution has several immediate drawbacks. In this case, each sensor requires knowledge about the centralized filter implementation which effectively reverses the original modularization problem. This solution also potentially creates a communications bandwidth issue, as the centralized filter must request predicted measurements from each sensor. As a result, the design prefers the solution of dynamic loading, although this software implementation detail is not immediately required by the system simulation in Chapter 4. Based on the previous discussion, it is apparent that the modularized architecture satisfies the majority of the desired filter properties.

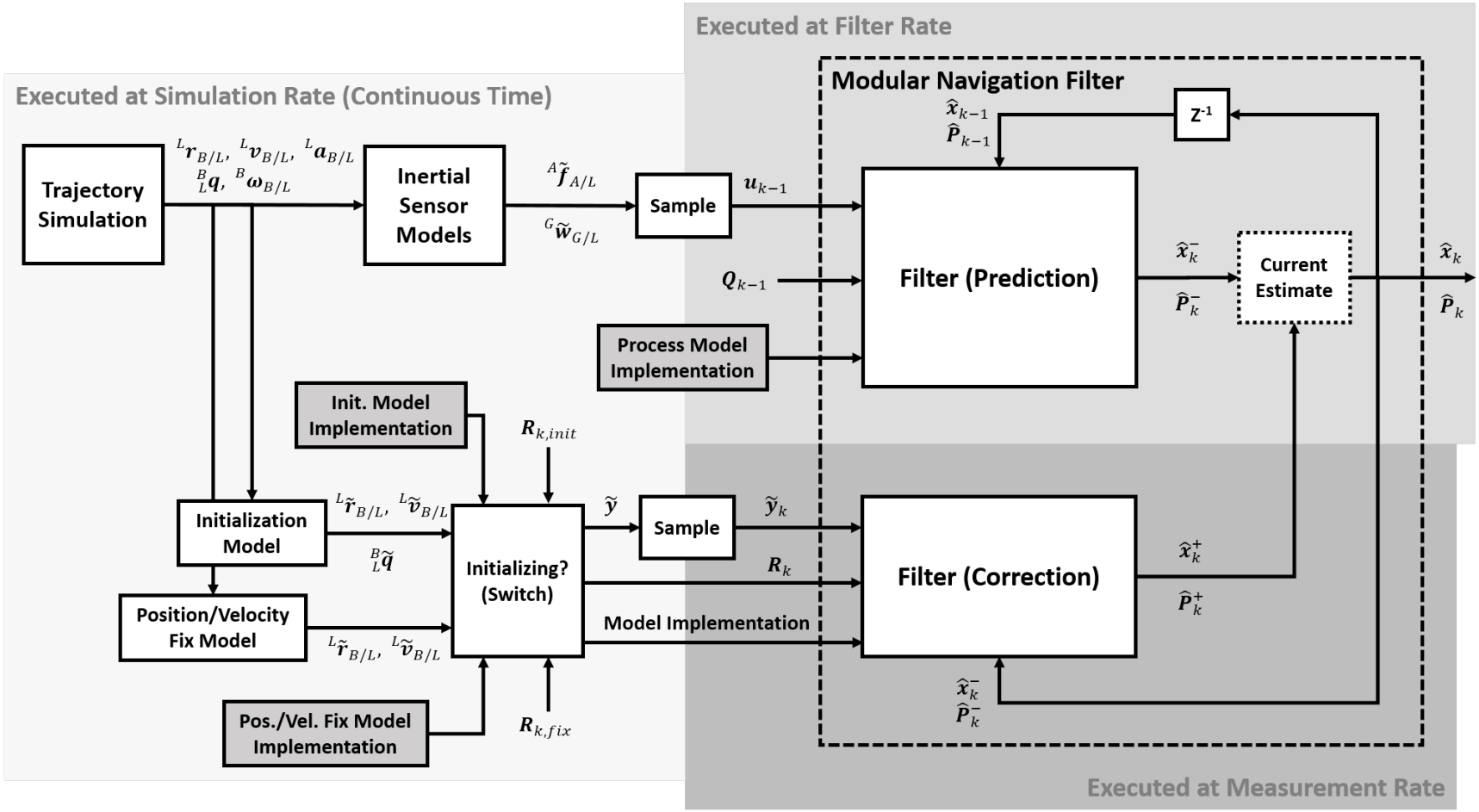

Figure 3.3: Simulation Architecture

The next step is to introduce the overall simulation architecture. This introduction provides top-level context for each of the filter components detailed in the following sections. The simulation architecture is shown in Figure 3.3. This figure depicts the modular navigation filter applied in the context of a computer simulation. The filter itself processes simulated inertial and aiding sensor measurements generated from simulated trajectory data. Inputs and outputs are generalized as much as possible to maintain flexibility. The following sections discuss the development and implementation of each of these components.

3.2 Filter Selection

The filter design process would typically proceed with development of the system models to inform filter selection. For this design, overall architecture flexibility and system performance are driven largely by the choice of the central filter type under the basic assumption of reasonable model nonlinearity, rather than the specific model definitions. For this reason, the filter selection is performed before initial model development. For context, this phase of the design refers to selecting an algorithm to perform the filter prediction and correction functions shown in Figure 3.3. The EKF is widely considered the standard filter for navigation applications due to its relative ease of use and proven performance [48]. However, the EKF falls short of the design guidelines in one important respect. The EKF makes use of analytically derived derivatives of the process and measurement model equations to perform a first-order linearization prior to propagating the state estimate distribution [28]. The use of these derivatives introduces design inflexibility due to the required re-derivation for every change to the system models, as well as a configuration management burden for platforms with a variety of aiding sensors. Among the studied filters, the EKF was rejected for these reasons.

The UKF provides convenient solutions to these problems. The second or higher-order approximation afforded by the UKF offers an improvement compared to the EKF. Citing numerous successful implementations in land and air navigation systems, this approxima-

tion is sufficient for a first iteration of the design [26, 49, 50, 51]. Since only a relatively small number of sigma points are required for an accurate approximation, the UKF demonstrates superior computational efficiency to the PF for a general application. At the same time, however, the UKF lacks the ability to accurately approximate multimodal state distributions [52]. In an effort to balance computation and performance, it is sufficient for now to assume the state distribution will remain nearly unimodal and symmetric about the mean. Multimodality is usually the result of aiding sensor measurements which contradict the current state estimate [53]. A variety of measurement rejection policies exist to alleviate this issue, such as the Chi-squared test [19]. Any deviation from these assumptions will be addressed in subsequent design iterations.

Another key property of the UKF lies in its direct use of the system model equations to perform any required transformation. Due to this direct use, no model derivatives are required and no explicit linearization is performed. This last fact allows for the UKF mechanization to remain abstract with respect to model details, minimizing the effects of modifying the system models. In this sense, the UKF uses the minimum possible model information for state estimation, a property shared with the PF. This commonality is also beneficial since it allows the PF to easily replace the UKF in the design should a future need arise to handle multimodal or non-symmetric distributions. Based on these properties, the UKF balances the present need for flexibility and the amount of required computation, and is the selected filter for this design. Commonality with the PF also facilitates the progression to a SLAM algorithm should the future need arise.

There is one last design consideration to note at this point in regard to future expansion. For a lidar-aided inertial system, there are two possible implementations described in Section 2.4.3. The first implementation produces localization estimates by comparing lidar data collected in real-time with a known dataset, or point cloud, collected in the past. This implementation is properly handled by the proposed architecture, as data collected in real-time are related to the prior dataset to estimate the system state. In this case, there is no

additional correlation between the online measurements and the prior data. This is not true, however, for the second case, which involves simultaneous mapping as in SLAM. In this second case, features in the map are heavily correlated with the state estimate, and the problems of state estimation and mapping are no longer independent [51, 54]. This case must be approached from the perspective of SLAM where feature locations become part of the system state. The present design focuses on the former case and simply attempts to achieve sufficient flexibility to permit future growth toward supporting SLAM. With the selection of the central filter, the design may proceed to define the initial process and measurement models.

3.3 System Model Design

The system model design influences multiple elements in the system architecture diagram (Figure 3.3), but the focus of this design phase is the process model implementation. The top-level process model is depicted in Figure 3.4. This research employs the standard "model-less" approach to the process model design as described in [55, 26]. The system model architecture centers around an IMU providing reliable acceleration and angular rate data. These measurements are inserted directly into the kinematics model of the tracked object. In this configuration, the process model assumes no additional knowledge of the vehicle or platform dynamics. As noted by Christophersen, et al, in [55], the direct use of the inertial sensor measurements in the process model accounts for the process noise input.

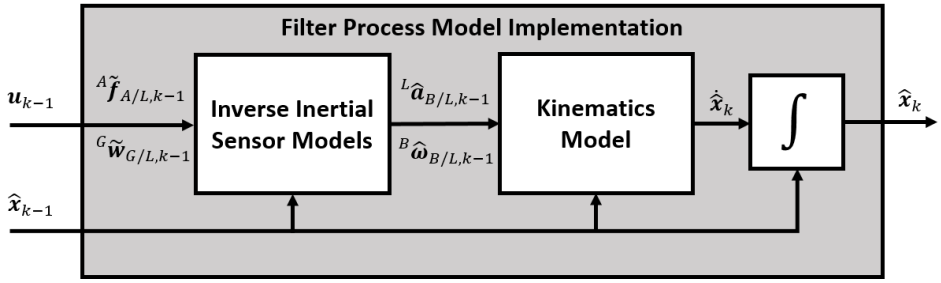

Figure 3.4: Process Model Implementation

This approach preserves applicability to a wide variety of operating environments at the cost of the potential performance improvement gained by considering more detailed knowledge of the system dynamics. In the typical fashion, the filter handles updates from aiding sensors such as GNSS through the measurement update. The next few sections discuss development of the process model components in Figure 3.4.

3.3.1 Coordinate Systems

Prior to actually defining the process model, an appropriate coordinate system must be chosen. Several coordinate systems see widespread popularity for navigation systems depending on the application. Some common systems are the earth-centered inertial (ECI), earth-centered earth-fixed (ECEF), and the local geodetic frame (LGF) [10]. The choice of coordinate frame drastically alters the form of the system kinematics.

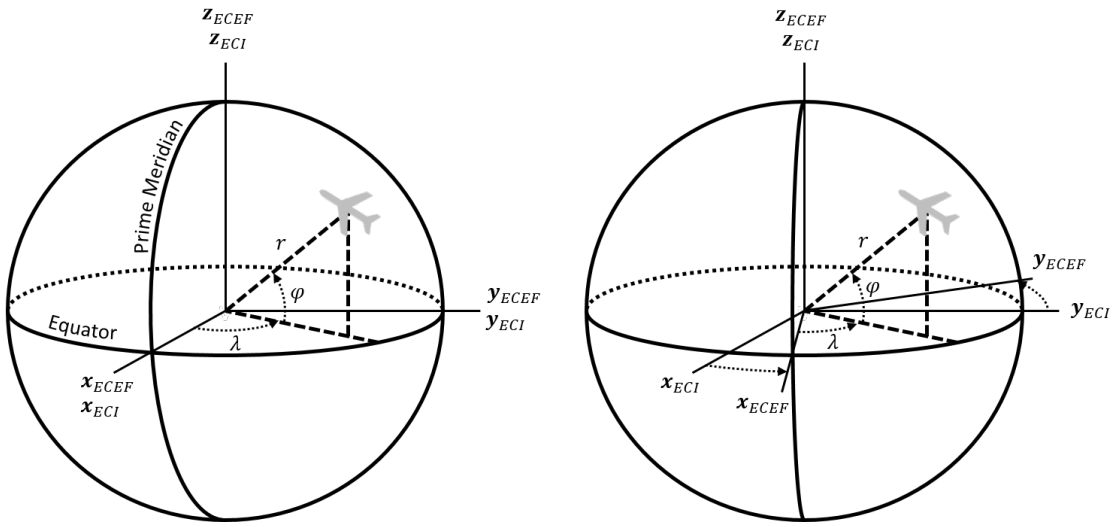

Figure 3.5: Global Coordinate Frames

The ECI frame is a non-rotating Cartesian coordinate frame with its origin at the center of the earth, represented by the coordinate axes $\boldsymbol{x}_{ECI}$, $\boldsymbol{y}_{ECI}$, and $\boldsymbol{z}_{ECI}$ in Figure 3.5. The

ECI frame does not rotate with the earth about its polar axis and, consequently, approximates a roughly inertial reference frame. As a result, the system kinematics resolved in the ECI frame follow standard Newtonian rules without the introduction of fictitious forces. However, as an approximation, the ECI frame neglects relativistic effects, precession of the earth's rotation, and small accelerations due to the earth's revolution around the sun. Still, the approximation is suitable for most terrestrial applications.

The ECEF frame, given by $\boldsymbol{x}_{ECEF}$, $\boldsymbol{y}_{ECEF}$, and $\boldsymbol{z}_{ECEF}$ in Figure 3.5, shares the same origin as ECI but rotates with the earth. This rotation creates the need to introduce fictitious forces in the kinematics to account for inertial effects. The relatively minimal additional complexity is often eclipsed by convenience since ECEF allows location on the earth to be resolved in a more relatable manner. LGF coordinates add intuition to this relatability. The LGF coordinate system represents points on the surface of the earth by their respective latitude (φ), longitude (λ), and altitude (h). Altitude is defined as the height above an ellipsoid that roughly models the surface of the earth, and is related to the orbital radius r. The World Geodetic System 1984 (WGS84) standard provides a common definition of this ellipsoid. LGF coordinates take an intuitive form drawn from basic cartography, but the involved coordinate projections introduce significant complication into the kinematic equations. It is more common to perform coordinate conversions to the LGF from another frame.

Other local coordinate frames exist as well that differ in complexity due to fictitious forces or coordinate projections. Despite the additional complexity, one advantage of a local coordinate system is the reduced arithmetic precision typically required for computation. As global-scale coordinates, ECEF and ECI coordinates usually require a double-precision floating-point representation for accurate storage and manipulation. Even with a GPU-accelerated target platform, double-precision arithmetic may run up to 32 times slower than single-precision, making double-precision unappealing for many embedded real-time applications [56]. Despite the potential performance impact, this research as-

sumes the availability of double precision arithmetic in order to preserve compatibility with a global-scale coordinate system in the future.

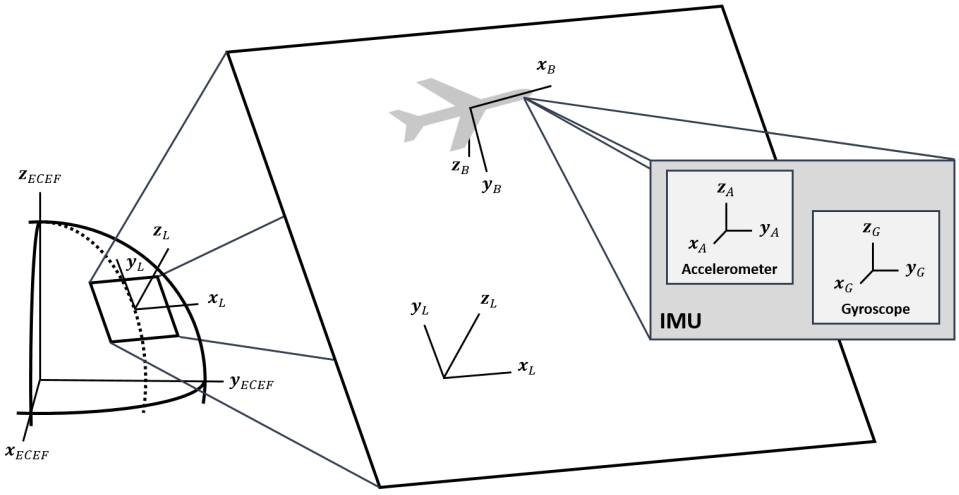

Figure 3.6: Local Coordinate Frame

For the initial design, however, the system kinematics are formulated using a simple local Cartesian coordinate system (L) centered at the location of system initialization. This sort of system is often known as a tangent-plane coordinate system [57]. Coordinate axis directions are defined based on the local level and cardinal directions, with $\boldsymbol{x}_L$ pointing in the East direction, $\boldsymbol{y}_L$ pointing toward true North, and $\boldsymbol{z}_L$ pointing in the local up direction, normal to the WGS84 ellipsoid. This frame is commonly known as the east-north-up (ENU) frame and is depicted in Figure 3.6. This simple coordinate system is assumed inertial for relatively small excursions of short duration, but fictitious forces due to the rotation of the Earth are not included in the kinematics. For additional simplicity in this design, the local frame does not translate or rotate with the body frame and remains fixed at initialization, causing the axes to become misaligned from the true cardinal directions over distance. Due to these assumptions, travel distance from the origin must be limited to about 150 kilometers to minimize the effects of earth's curvature and rotation. The actual error

due to these effects is a function of the excursion latitude, altitude, and overall distance. In a real application with GNSS aiding, a position fix and accurate heading reference would be required to correctly define the origin of this local ENU frame.

Several other non-inertial reference frames are defined within this system to represent the tracked object and sensor installations. Each reference frame is related to the local frame, L, through a common object body frame, B, by an independent translation and rotation. The body frame is rigidly fixed to the tracked object. For standard aircraft applications, the body frame is usually defined with $\boldsymbol{x}_B$ pointing toward the front of the aircraft, $\boldsymbol{y}_B$ pointing toward the right side, and $\boldsymbol{z}_B$ pointing down. This definition coincides with the common aircraft definition for right-handed heading, pitch, and roll rotations. Based on this definition, the object position in the local frame is simply the translational offset between the local and body coordinate systems. The object attitude, or orientation, is represented by the rotational offset between these reference frames. However, attitude is more commonly expressed as heading (ψ), pitch (θ), and roll (ϕ) which are measured with respect to true north and the local level defined by the WGS84 ellipsoid. To define this standard convention, the north-east-down (NED) coordinate frame is required, denoted by N. Like the ENU frame, the NED coordinate axes are typically fixed to the object body and rotate to remain aligned with the cardinal directions. For this design, the NED coordinate axes are assumed static in direction similar to the ENU frame. The standard yaw, pitch, and roll rotations are defined as the rotation of the body frame B with respect to NED frame.

The local accelerometer (A) and gyroscope (G) frames are defined with fixed translational offsets and rotations from the body frame, allowing vector measurements to be resolved in, or projected onto, any of the defined coordinate systems. To discuss these transformations in general, a notation and representation must be defined for three-dimensional rotations.

3.3.2 Coordinate Transformations

Many representations exist to express three-dimensional rotations. Common representations include DCMs, quaternions, axis-angle, and Euler angles. Despite the deceivingly unintuitive nature of rotations, the intricacies of each representation have been widely studied and their respective effects on the kinematic equations are well-understood [58, 59]. In lieu of the multitude of other references, this paper will only briefly discuss representations useful to the research at hand. This research will apply the Euler angle, DCM, and quaternion representations where appropriate to achieve the desired intuition in different contexts.

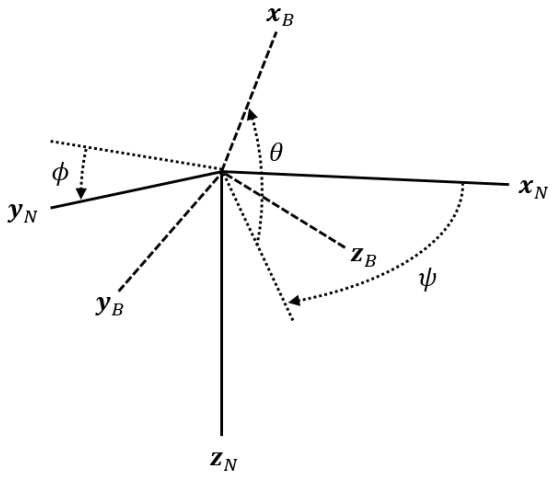

Figure 3.7: Tait-Bryan Angles

Euler angles provide the greatest intuition to visualize the attitude of an object. Three Euler angles are required to define a three-dimensional rotation and many Euler angle sequences exist which differ in the axis and order of rotations. Aerospace applications commonly employ the Tait-Bryan angles which represent the yaw (ψ), pitch (θ), and roll (ϕ) of an object. This rotation sequence is called the 3-2-1 sequence since rotations occur about the 3rd, 2nd, and 1st (or Z, Y, and X) axes, in that order. It is important to note that these

angles represent rotation of the coordinate axes rather than rotation of a vector within a coordinate system. In different fields, coordinate axis rotations are often also called passive or alias rotations [60]. Vector rotations, on the other hand, are known as active or alibi rotations [61]. In the Tait-Bryan sequence, yaw, pitch, and roll are measured with respect to the true north, east, and down directions, as depicted in Figure 3.7.

Despite their intuitive nature, Euler angles exhibit many deficiencies as a representation mainly due to singularities that occur when the rotation axes align. Also, any particular rotation has multiple Euler angle representations within the same rotation sequence which can lead to problems during interpolation between rotations. Therefore, the Tait-Bryan angles are only employed where readability or intuition are required. The 3-by-3 DCM provides a superior, although more verbose, representation. Elements in the DCM are derived from permutations of the inner products between coordinate system axes. As a result, DCMs are orthonormal matrices. The DCMs for the Tait-Bryan rotation sequence are given in Equation 3.1. The primary benefit of the DCM is the intuition it provides for the order of rotations and operations on vectors. Using the Tait-Bryan sequence, the rotation of a vector resolved in coordinate system A to a coordinate system B occurs as in Equation 3.2.

$$
\begin{aligned}
\boldsymbol{T}_z(\psi) &= \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
\boldsymbol{T}_y(\theta) &= \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \\
\boldsymbol{T}_x(\phi) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}
\end{aligned}
\tag{3.1}
$$

$$^{B}\boldsymbol{v} = {}^{B}_{A}\boldsymbol{T}(\psi,\theta,\phi)\,{}^{A}\boldsymbol{v} = {}^{B}_{A_{zy}}\boldsymbol{T}_x(\phi)\,{}^{A_{zy}}_{A_z}\boldsymbol{T}_y(\theta)\,{}^{A_z}_{A}\boldsymbol{T}_z(\psi)\,{}^{A}\boldsymbol{v} \tag{3.2}$$

In Equation 3.2, the matrix ${}^{B}_{A}\boldsymbol{T}(\psi,\theta,\phi)$ represents the combined Tait-Bryan sequence of the three-axis rotation. In this form, the DCM provides an intuitive way to view cascaded coordinate transformations operating on a vector or another DCM. The prefixed matrix product notation in Equation 3.2 also provides an expressive way to describe which rotation a given DCM represents and in which reference frame a vector is resolved. The notation in Equation 3.2 requires diagonal prefix terms to match for a valid rotation sequence. Since DCMs are orthogonal, the inverse of a rotation is computed by transposing the matrix, corresponding to a swap of prefix terms. Although verbose, this notation is a useful accounting mechanism when many coordinate systems are simultaneously used.

The DCM does not share the same limitations as Euler angles but requires the storage of nine parameters instead of three. For some applications, this storage requirement is not prohibitive since computer memory is relatively inexpensive. For other applications, unit quaternions are a better solution. Requiring only four parameters, unit quaternions offer higher information density than the DCM and avoid the pitfalls of Euler angles. Since only three parameters are required to fully define a rotation, the four parameters are constrained to unit magnitude to effectively remove the extra degree of freedom. Unit quaternion rotations are not fully unique in that additive inverses represent the same rotation. However, due to the geometric separation of these vectors, smooth interpolation between rotations can be achieved, a property that is useful for upsampling low-rate navigation data. The quaternion attitude kinematics assume a relatively simple and straightforward form, comparable to that of the DCM. This paper makes use of unit quaternions for representing attitude, but uses the familiar DCM pre-multiply for vector rotation to capitalize on intuition. In these cases,

the DCM is denoted as $\boldsymbol{T}(\boldsymbol{q})$ where $\boldsymbol{q}$ is the unit quaternion attitude. This conversion is given by Equation 3.3 where the four quaternion parameters are q_x, q_y, q_z, and q_s.

$$\boldsymbol{T}(\boldsymbol{q}) = \begin{bmatrix} q_s^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y + q_z q_s) & 2(q_x q_z - q_y q_s) \\ 2(q_x q_y - q_z q_s) & q_s^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z + q_x q_s) \\ 2(q_x q_z + q_y q_s) & 2(q_y q_z - q_x q_s) & q_s^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{3.3}$$

As the second component of a transformation, translations between coordinate systems require knowledge of the relative location of each coordinate system origin. A similar notation is available to manage order and validity of translations. A pure translation of a vector from A to B resolved in B is given in Equation 3.4.

$$^B\boldsymbol{r}_B = {}^B\boldsymbol{r}_{B/A} + {}^B\boldsymbol{r}_A \tag{3.4}$$

The translation in Equation 3.4 essentially relocates the origin for the vector, ${}^B\boldsymbol{r}_A$, from A to B using knowledge of the vector distance from A to B, ${}^B\boldsymbol{r}_{B/A}$. For a valid translation, each vector must be resolved in the same reference frame and the offset vector must represent a distance from the origin of ${}^B\boldsymbol{r}_A$ to some other origin. These conditions are reflected by matching prefix superscripts and cascading postfix subscripts, respectively. Assuming all vectors are resolved in the same coordinate system, the translation equation in Equation 3.4 can be differentiated to determine relative velocity and acceleration. An important result from basic dynamics is that this differentiation yields additional terms when one of the coordinate systems is in motion, as is the case for the ECEF coordinate frame rotating with the Earth [62]. Using the so-called Coriolis theorem, the velocities and accelerations in the inertial (S) and rotating (R) coordinate systems are related in Equation 3.5 and Equation 3.6. These additional terms correspond to the fictitious forces that occur in non-inertial reference frames. In the present case, these terms arise due to the translational offset of the accelerometer from the body frame and the rotation of the Earth.

$$\dot{\boldsymbol{r}}_{P/S} = \dot{\boldsymbol{r}}_{R/S} + \dot{\boldsymbol{r}}_{P/R} + \boldsymbol{\omega}_{R/S} \times \boldsymbol{r}_{P/R} \tag{3.5}$$

$$\ddot{\boldsymbol{r}}_{P/S} = \ddot{\boldsymbol{r}}_{R/S} + \ddot{\boldsymbol{r}}_{P/R} + \dot{\boldsymbol{\omega}}_{R/S} \times \boldsymbol{r}_{P/R} + 2\,\boldsymbol{\omega}_{R/S} \times \dot{\boldsymbol{r}}_{P/R} + \boldsymbol{\omega}_{R/S} \times (\boldsymbol{\omega}_{R/S} \times \boldsymbol{r}_{P/R}) \tag{3.6}$$

$$\boldsymbol{\omega}_{P/S} = \boldsymbol{\omega}_{P/R} + \boldsymbol{\omega}_{R/S} \tag{3.7}$$

Angular rate vectors, denoted by $\boldsymbol{\omega}$, describe the rotation rates of a coordinate system about their respective axes, following the same rules for vector translation. Conveniently, the angular rate transform equation in Equation 3.7 shows that an additive relationship between angular rates is preserved for a rotating coordinate system and no additional terms are required. The ECEF reference frame kinematics are essentially derived from these equations with the addition of the appropriate inertial measurements. For the simplified local navigation frame used in this paper, additional terms related to the Earth's rotation are neglected. The accelerometer model, however, makes use of the Coriolis theorem in Equation 3.6 to include cases where the accelerometer is offset in translation from the body axis origin. In the literature, these transformations and the resulting kinematic equations are often expressed in matrix form using multiplication with skew-symmetric matrices instead of vector cross products. With the required coordinate systems and transformations defined, the design continues with the development of the system kinematics for the model-less process equations.

3.3.3 Kinematics

The system kinematics follow from standard Newtonian physics. The system process model is defined in Equation 3.8 according to the standard convention used in this paper. The control input term $\boldsymbol{u}$ is neglected in the model-less approach since no specifics

of the platform dynamics are known. It is instead used as an input for the inertial sensor measurements. This usage is depicted in Figure 3.4.

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{w}(t))) \tag{3.8}$$

The system state $\boldsymbol{x}$ consists of the platform position ${}^{L}\boldsymbol{r}_{B/L}$, velocity ${}^{L}\boldsymbol{v}_{B/L}$, accelerometer bias ${}^{A}\overline{\boldsymbol{f}}_{A/L}$, attitude ${}^{B}_{L}\boldsymbol{q}$, and angular rate bias ${}^{G}\overline{\boldsymbol{\omega}}_{G/L}$. Each quantity is represented as a 3-by-1 vector with the exception of attitude, which is represented as a 4-by-1 unit quaternion. The position and velocity vectors are resolved in the local frame (L). The position vector represents the displacement from the origin of the navigation frame to the origin of the platform body frame. The velocity vector represents the velocity of the body frame origin relative to the navigation frame origin. The attitude quaternion ${}^{B}_{L}\boldsymbol{q}$ represents the rotation from the navigation frame to the body frame. The accelerometer and angular rate bias vectors are resolved in their respective sensor frames A and G. These bias vectors arise from the inertial sensor models which are discussed later in Section 3.4. The complete system state is captured in the 16-by-1 vector $\boldsymbol{x}$ shown in Equation 3.9.

$$\boldsymbol{x} = \begin{bmatrix} {}^{L}\boldsymbol{r}_{B/L} \\ {}^{L}\boldsymbol{v}_{B/L} \\ {}^{A}\overline{\boldsymbol{f}}_{A/L} \\ {}^{B}_{L}\boldsymbol{q} \\ {}^{G}\overline{\boldsymbol{\omega}}_{G/L} \end{bmatrix} \tag{3.9}$$

$$\boldsymbol{u} = \begin{bmatrix} {}^{A}\widetilde{\boldsymbol{f}}_{A/L} \\ {}^{G}\widetilde{\boldsymbol{\omega}}_{G/L} \end{bmatrix} \tag{3.10}$$

Measurements are stored in the input vector, $\boldsymbol{u}$, defined in Equation 3.10. These measurements are described in more detail in Section 3.4. The process model itself follows from the derivatives of each state. The position and velocity derivatives, given by Equa-

tion 3.11 and Equation 3.12, are almost trivial in that they derive from the velocity state and acceleration measurement almost directly.

$$ {}^{L}\dot{\boldsymbol{r}}_{B/L} = {}^{L}\boldsymbol{v}_{B/L} \tag{3.11} $$

$$ {}^{L}\dot{\boldsymbol{v}}_{B/L} = {}^{L}\boldsymbol{a}_{B/L} \tag{3.12} $$

It is important to note any nonlinear coupling that occurs between states in these equations. Nonlinear coupling is not an issue by itself since the UKF is a nonlinear estimator. However, nonlinear coupling of stochastic inputs with system states violates one of the key assumptions of the UKF as presented in this research. In the velocity derivative, Equation 3.12, significant coupling is hidden by the term ${}^{L}\boldsymbol{a}_{B/L}$. This term is input from the accelerometer model, described in Section 3.4.2. Nonlinear coupling of stochastic inputs requires a state augmentation approach to correctly account for nonlinear effects. The form of the quaternion rotation matrix is another such source of coupling. The quaternion derivative is shown in Equation 3.13.

$$ \boldsymbol{\Omega}({}^{B}\boldsymbol{\omega}_{B/L}) = \begin{Bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{Bmatrix} \tag{3.13} $$

$$ {}^{B}_{L}\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\Omega}({}^{B}\boldsymbol{\omega}_{B/L})\,{}^{B}_{L}\boldsymbol{q} $$

$$ ||\,{}^{B}_{L}\boldsymbol{q}||_2 = 1 \tag{3.14} $$

This derivative is a standard result achieved through perturbation analysis of the derivative formula for the quaternion [63]. This result is sometimes recast as a transformation of the angular rate vector, although the forms are equivalent. The quaternion ${}^{B}_{L}\boldsymbol{q}$ is also con-

strained to unit magnitude, via the l_2 norm, as in Equation 3.14. Artificial implementation of this constraint through renormalization is more or less a brute force method to ensure the constraint is satisfied. Many other successful techniques exist to more naturally ensure this constraint, such as propagating attitude error states rather than the full attitude quaternion [64]. This technique includes an added benefit in that only three attitude error states are required, reducing the overall number of states. Furthermore, an error state derivation is available for the entire process model which can reduce the nonlinear model to a linear form [22]. The error state approach was not implemented in any capacity for the initial design of the process model for the sake of simplicity. It is important to note that the order of terms in the matrix $\boldsymbol{\Omega}({}^{B}\boldsymbol{\omega}_{B/L})$ is determined by the order of the quaternion parameters. The order given in Equation 3.13 is valid when the scalar value is the last quaternion parameter, as in the vector $[q_x, q_y, q_z, q_s]^{\mathsf{T}}$. Similar to the acceleration in Equation 3.12, the angular rate vector ${}^{B}\boldsymbol{\omega}_{B/L}$ assumes a simple form representing the output of the gyroscope model, described in Section 3.4.1. Both of these quantities are internally dependent on the state estimate and introduce further cross coupling into the system. Both quantities also contain measurement inputs which introduce process noise.

The final two values of the process model are the accelerometer and gyroscope biases. As previously mentioned, these quantities originate from the inertial sensor models and are further discussed in Section 3.4. For now, it is sufficient to describe them as variables undergoing random walk. Their derivatives are defined using the simple random walk model as in Equation 3.15 and Equation 3.16.

$$\begin{aligned} {}^{A}\dot{\overline{\boldsymbol{f}}}_{A/L} &= {}^{A}\boldsymbol{w}_{\dot{\overline{\boldsymbol{f}}}} \\ {}^{A}\boldsymbol{w}_{\dot{\overline{\boldsymbol{f}}}} &\sim \boldsymbol{N}(0, \boldsymbol{\sigma}^2_{\dot{\overline{\boldsymbol{f}}}}) \end{aligned} \tag{3.15}$$

$$\begin{aligned} {}^{G}\dot{\overline{\boldsymbol{\omega}}}_{G/L} &= {}^{G}\boldsymbol{w}_{\dot{\overline{\boldsymbol{\omega}}}} \\ {}^{G}\boldsymbol{w}_{\dot{\overline{\boldsymbol{\omega}}}} &\sim \boldsymbol{N}(0, \boldsymbol{\sigma}^2_{\dot{\overline{\boldsymbol{\omega}}}}) \end{aligned} \tag{3.16}$$

The model equations discussed to this point have been presented in their ideal form. These equations are useful for the trajectory simulation component of Figure 3.3 where specific values of noise are known. The filter itself must rely only on the inertial sensor measurements and noise statistics, and thus, specific realizations of the stochastic inputs must be omitted. The complete filter process model equations are shown in Equation 3.17, Equation 3.18, Equation 3.19, Equation 3.20, Equation 3.21, and Equation 3.22.

$$ {}^{L}\dot{\hat{\boldsymbol{r}}}_{B/L} = {}^{L}\hat{\boldsymbol{v}}_{B/L} \tag{3.17} $$

$$ {}^{L}\dot{\hat{\boldsymbol{v}}}_{B/L} = {}^{L}\hat{\boldsymbol{a}}_{B/L} \tag{3.18} $$

$$ {}^{A}\dot{\overline{\hat{\boldsymbol{f}}}}_{A/L} = \mathbf{0} \tag{3.19} $$

$$ \boldsymbol{\Omega}({}^{B}\hat{\boldsymbol{\omega}}_{B/L}) = \begin{Bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{Bmatrix} \tag{3.20} $$

$$ {}^{B}_{L}\dot{\hat{\boldsymbol{q}}} = \frac{1}{2}\boldsymbol{\Omega}({}^{B}\hat{\boldsymbol{\omega}}_{B/L})\,{}^{B}_{L}\hat{\boldsymbol{q}} $$

$$ ||\,{}^{B}_{L}\hat{\boldsymbol{q}}||_2 = 1 \tag{3.21} $$

$$ {}^{G}\dot{\overline{\hat{\boldsymbol{\omega}}}}_{G/L} = \mathbf{0} \tag{3.22} $$

These equations comprise the kinematics model component of Figure 3.4. The key difference in these equations is the lack of process noise terms, since these values are unknown in real time. This omission is more apparent for the bias estimate states, but also exists in the inertial sensor models as well. Knowledge of process noise statistics is incorporated through the process noise covariance, $\boldsymbol{Q}$. The next step to complete this process model is to perform discretization.

3.3.4 Discretization

The continuous time model given in Equation 3.8 must undergo discretization since the filter architecture is formulated for a discrete-time model. This operation is shown by the integrator component in Figure 3.4. There are a number of schemes for discretization that achieve varying levels of approximation error, such as those covered briefly in Section 2.3.3. From a qualitative standpoint, an appropriate discretization scheme should capture dynamics of at least the same order as the system [51]. The current kinematic model is of the second order due to the presence of acceleration and, therefore, a second-order discretization is used in this design. The modified Heun method performs the required operation [24]. This two-step method was selected over other second-order methods because, unlike the more familiar Milstein approximation, it does not require any model derivatives and it reduces to the common explicit trapezoidal method when the stochastic inputs are zero. These appealing features are attained at the cost of the extra computation step. Given a stochastic differential equation of the form in Equation 3.23, the process in Equation 3.24 discretizes the continuous-time equation.

$$\dot{x} = f(t, x(t))dt + g(t, x(t))dw \tag{3.23}$$

$$\begin{aligned} K_{1,k} &= f(t_k, x_k) + (\Delta w_k - s_k)\frac{g(t_k, x_k)}{\sqrt{T_s}} \\ K_{2,k} &= f(t_k + T_s, x_k + K_{1,k}) + (\Delta w_k + s_k)\frac{g(t_k + T_s, x_k + K_{1,k})}{\sqrt{T_s}} \\ x_{k+1} &= x_k + \frac{1}{2}T_s(K_{1,k} + K_{2,k}) \end{aligned} \tag{3.24}$$

In Equation 3.24, Δw_k signifies a zero-mean white-noise input with a variance of one and s_k is sampled uniformly from the set $\{-1, 1\}$. It is clear from these equations that when the stochastic volatility function g is zero, Equation 3.24 assumes the form of the simple explicit trapezoidal, or improved Euler, method. This simplification occurs for the filter

process model itself, but the full discretization is required to simulate the process model. This method provides integration error that is linear in the time step T_s. Simulated noise must be scaled appropriately to ensure the input power does not change with the time step. This scaling is captured by the $\sqrt{T_s}$ in the denominator of Equation 3.24.

In the standard KF framework, the process model noise $\boldsymbol{w}$ is specified as zero-mean with a covariance $\boldsymbol{Q}$. It is necessary to scale $\boldsymbol{w}$ by $1/\sqrt{T_s}$ to accommodate the noise scaling given by the integration framework in Equation 3.24. However, simple scaling does not account for the random perturbation caused by the uniform distribution of s_k. For the purposes of this research, the additional stochastic term was neglected since the resulting error scales with the magnitude of the noise and the square root of the time step. Furthermore, the kinematic process model does not share the form of Equation 3.23 since the process noise is not purely additive. These inconsistencies are potential causes of integration error for a large time step or if the process model becomes highly nonlinear.

3.3.5 Process Noise

The one remaining discrepancy between the current form of the model and the form expected by the standard UKF is the assumption of additive process noise. For now, it is assumed that coupled noise is accounted by adding zero-mean, uncorrelated, white noise to each state. This additive noise must account for all model errors, including inertial measurement and discretization errors. This noise also provides the ability to express generic uncertainty in the process model. Ideally, the covariance of this process noise is derived from that of the actual stochastic inputs. In theory, it is possible to express the covariance in terms of the sensor noise covariance and worst-case discretization error. However, in practice, the covariance must be tuned to correctly account for these errors and other model uncertainties.

The general process noise covariance for this design is given by Equation 3.25. An important distinction is that none of these process noise variances directly represent any of

the measurement noise variances defined in this chapter due to the discretization step and other nonlinearities in the model. It is possible to derive the relationship between these values to compute the process noise variances analytically, but this analysis was omitted for now in favor of using a rough approximation and subsequent manual tuning.

$$\boldsymbol{Q}_k = \begin{bmatrix} \boldsymbol{\sigma}_{\boldsymbol{r}}^2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\sigma}_{\boldsymbol{v}}^2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma}_{\bar{\boldsymbol{f}}}^2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma}_{\boldsymbol{q}}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma}_{\bar{\boldsymbol{\omega}}}^2 \end{bmatrix} \tag{3.25}$$

3.4 Inertial Sensor Modeling

The next central filter components are the inertial sensor models, shown in Figure 3.3. These models capture the effects of random walk, bias, and scale factor errors while accounting for translational and rotational installation offsets. The coordinate systems for these offsets were previously defined in Section 3.3.1. These inertial sensor models serve as a baseline for future development and specialization since each inertial sensing technology is prone to different kinds of errors. The discussion of sensor modeling begins with the angular rate gyroscope model due to its relative simplicity. The accelerometer model naturally follows from a similar definition with some added complexity due to the kinematics of rotating reference frames. The sensor models presented in this section have been proven adequate for many types of IMU [10, 65, 66].

3.4.1 Angular Rate Gyroscope Model

The angular rate gyroscope model describes a noisy sensor measurement in the gyroscope frame, ${}^{G}\widetilde{\boldsymbol{\omega}}_{G/L}$, in terms of the true angular rate of the system resolved in the body frame, ${}^{B}\boldsymbol{\omega}_{B/L}$. The angular rate model is shown in Equation 3.26 and Equation 3.27. There

are two primary sources of measurement error in this model. The first source, known as angle random walk (ARW), is best interpreted as additive, zero-mean, white noise. ARW is usually the result of electrical noise and vibration in the sensor [67]. Since ARW is modeled as white noise, its contribution tends to decrease when measurements are averaged over time. ARW is denoted by ${}^{G}\boldsymbol{w}_{G}$ in Equation 3.27. Values for ARW are sampled from a normal distribution with variance $\boldsymbol{\sigma}^2_{\boldsymbol{w}_G}$. For the purpose of simulation, this variance is computed from the corresponding sensor specifications which are typically provided as a function of sensor bandwidth, with units of $\frac{degrees}{hour\sqrt{Hz}}$.

The second source of error, known as gyroscope bias offset error, is of much greater concern. Bias offset error, represented in Equation 3.27 by ${}^{G}\overline{\boldsymbol{\omega}}_{G/L}$, is manifest when the zero crossing of the sensor output occurs for a nonzero input. While it is possible to measure and effectively remove the bias offset, this removal is complicated by the fact that the offset changes over time. This change is known as bias instability. Bias offset and instability thus contribute to errors in long-term averages which will cause the estimated state to drift. Since this bias is not truly constant, it is necessary to estimate its value over time. As a result, the bias offset is included in the process model as an additional system state, as mentioned in Section 3.3.3. For simulation, the bias offset is modeled as a random walk process driven by zero-mean white noise ${}^{G}\boldsymbol{w}_{\dot{\overline{\boldsymbol{\omega}}}}$ sampled from a normal distribution with variance $\boldsymbol{\sigma}^2_{\dot{\overline{\boldsymbol{\omega}}}}$. This variance is computed from the manufacturer-specified bias instability often given in units of $\frac{degrees}{hour}$. The model for bias instability was shown previously with the process model in Equation 3.16.

The first step of the angular rate model is to transform the body-resolved angular rate vector to the gyroscope frame, depicted in Equation 3.26. As previously mentioned, the version of the transformation used in this research does not model the effect of Earth's rotation. The second equation, Equation 3.27, gives the actual sensor model.

$$^{G}\boldsymbol{\omega}_{G/L} = {}^{B}_{G}\boldsymbol{T}^{\mathsf{T}}\,{}^{B}\boldsymbol{\omega}_{B/L} \tag{3.26}$$

$$^{G}\widetilde{\boldsymbol{\omega}}_{G/L} = {}^{G}\overline{\boldsymbol{\omega}}_{G/L} + {}^{G}\boldsymbol{w}_{G} + (\boldsymbol{I} + \boldsymbol{M}_{G})\,{}^{G}\boldsymbol{\omega}_{G/L} \tag{3.27}$$

This model also includes two lesser sources of error, namely, scaling and cross-coupling errors, represented by the 3-by-3 matrix $\boldsymbol{M}_G$ in Equation 3.27. These errors affect sensitivity to the measured quantities along the measurement axes of the sensor. Scaling errors appear as inconsistent, sometimes nonlinear, sensitivity over the full scale range of the measured quantity. Cross-coupling errors occur when a measurement axis is sensitive to motion that should only appear on the other axes, potentially because the axes are not perfectly orthogonal. These sources of error are usually insignificant compared to random walk and bias offset errors. While estimation of these quantities is possible, some studies have shown that the simple white noise and bias model is sufficient to handle these effects [26]. For this research, random scaling errors are included in simulated sensor measurements to test robustness, but these errors are not estimated by the filter.

As a final note on the subject, there are many other sources of inertial sensor error that arise from other practical considerations. Some inertial sensor errors are temperature dependent. In these cases, it is possible to characterize this dependence and apply compensation as a function of measured temperature. MEMS and some electrolytic tilt sensing technologies apply this sort of compensation. Temperature and other additional sources of error are neglected for the purposes of this research.

The angular rate model in Equation 3.27 is useful for the purposes of inertial sensor measurement simulation, shown in Figure 3.3, but its effective inverse is required for use with the filter process model in Equation 3.20. This inverse model is depicted in Figure 3.4. The inverse is acquired by solving for $^{B}\boldsymbol{\omega}_{B/L}$ in Equation 3.27 and excluding noise terms which are unknown during real-time operation. The equation is rewritten in Equation 3.28

with proper annotations to distinguish measurements and estimates from their corresponding true values produced by the sensor model in simulation.

$$ {}^{B}\hat{\boldsymbol{\omega}}_{B/L} = {}^{B}_{G}\boldsymbol{T}(\boldsymbol{I} + \hat{\boldsymbol{M}}_{G})^{-1}[{}^{G}\widetilde{\boldsymbol{\omega}}_{G/L} - {}^{G}\hat{\overline{\boldsymbol{\omega}}}_{G/L}] \tag{3.28} $$

The inverted model equation in Equation 3.28 includes a matrix inverse of the quantity $(\boldsymbol{I} + \hat{\boldsymbol{M}}_{G})$. Some literature recommends approximation of this quantity through a power-series expansion [10]. Since this research neglects scaling and cross-coupling errors, the inverse reduces to the identity matrix. The resulting angular rate estimate ${}^{B}\hat{\boldsymbol{\omega}}_{B/L}$ is used directly by the quaternion propagation equation in the filter process model, given in Equation 3.20. Since values for the noise ${}^{G}\boldsymbol{w}_{G}$ are unknown during operation, the filter process dynamics for the bias estimate ${}^{G}\hat{\overline{\boldsymbol{\omega}}}_{G/L}$ are assumed constant, as in Equation 3.22.

3.4.2 Accelerometer Model

The accelerometer model, given in Equation 3.29, Equation 3.30, and Equation 3.31, is strikingly similar to that of the gyroscope. Accelerometers are similarly subject to noise, bias instability, scaling, and cross-coupling errors. Zero-mean white noise in the accelerometer model, ${}^{A}\boldsymbol{w}_{A}$, is known instead as velocity random walk (VRW). Its variance, $\boldsymbol{\sigma}^{2}_{\boldsymbol{w}_{A}}$, is usually provided with units of $\frac{g}{\sqrt{Hz}}$ where g refers to the nominal value of gravitational acceleration rather than the unit of grams. The bias offset for the accelerometer is also defined by a random walk model, shown in Equation 3.15. The white noise that drives bias instability, ${}^{A}\boldsymbol{w}_{\dot{\boldsymbol{f}}}$, is defined by a variance, $\boldsymbol{\sigma}^{2}_{\dot{\boldsymbol{f}}}$, and specified in units of g's. Scaling and cross-coupling errors are again represented by a matrix, denoted here as $\boldsymbol{M}_{A}$. Additional complexity in the accelerometer model comes from the coordinate frame transformation in Equation 3.29 and the specific force measurement model in Equation 3.30. Note that the transformation, ${}^{B}_{L}\boldsymbol{T}$, is computed from the quaternion attitude, ${}^{B}_{L}\boldsymbol{q}$, using the relationship in Equation 3.3.

$$ {}^{L}\boldsymbol{a}_{A/L} = {}^{L}\boldsymbol{a}_{B/L} + {}^{B}_{L}\boldsymbol{T}^{\mathsf{T}}[{}^{B}\dot{\boldsymbol{\omega}}_{B/L} \times {}^{B}\boldsymbol{r}_{A/B} + {}^{B}\boldsymbol{\omega}_{B/L} \times ({}^{B}\boldsymbol{\omega}_{B/L} \times {}^{B}\boldsymbol{r}_{A/B})] \tag{3.29} $$

$$ {}^{A}\boldsymbol{f}_{A/L} = {}^{B}_{A}\boldsymbol{T}^{\mathsf{T}}\,{}^{B}_{L}\boldsymbol{T}[{}^{L}\boldsymbol{a}_{A/L} - {}^{L}\boldsymbol{\gamma}({}^{L}\boldsymbol{r}_{B/L})] \tag{3.30} $$

$$ {}^{A}\widetilde{\boldsymbol{f}}_{A/L} = {}^{A}\overline{\boldsymbol{f}}_{A/L} + {}^{A}\boldsymbol{w}_{A} + (\boldsymbol{I} + \boldsymbol{M}_{A})\,{}^{A}\boldsymbol{f}_{A/L} \tag{3.31} $$

The angular rate transform equation, Equation 3.26, assumed a relatively simple form since rotating coordinate frames preserve the additive relationship of angular rates, as in Equation 3.7. However, rotation of the body frame with respect to the inertial frame causes fictitious forces to appear in the body frame acceleration. This transformation requires the use of the Coriolis theorem presented in Equation 3.6, and results in the equation given by Equation 3.29. This equation describes the fictitious forces, or additional accelerations, present when the accelerometer frame (A) is offset from the body frame in translation. One important term in this equation is the angular acceleration, ${}^{B}\dot{\boldsymbol{\omega}}_{B/L}$. This term is simple to determine in the simulation, but requires more care in handling when the model is inverted, since a direct measurement of angular acceleration is not usually available. It is also important to note that these additional terms disappear when the translational offset between the accelerometer and body frames, ${}^{B}\boldsymbol{r}_{A/B}$, is zero. One potential technique for avoiding fictitious forces is to move the body frame origin to the accelerometer frame, allowing the filter to track this location directly, effectively removing the extra terms. If necessary, the resulting state estimate can be later transformed to the desired body frame. The design in this research employs this technique for simplicity.

The specific force measurement transformation in Equation 3.30 complicates the accelerometer model as well. This equation arises from the fact that accelerometers measure the specific force, ${}^{A}\boldsymbol{f}_{A/L}$, rather than true acceleration, ${}^{A}\boldsymbol{a}_{A/L}$. Specific force differs from true inertial acceleration in that it lacks the gravitational acceleration component. To simulate specific force, the acceleration of gravity must be removed from the true acceleration

using an appropriate gravity model. For Earth-scale coordinate systems like ECEF, highly accurate gravity models are available which define the acceleration vector as a function of location. For the local coordinate system applied in this research, the gravity model is relatively simple, assuming a constant acceleration in the down direction ($-\boldsymbol{z}_L$). This model is defined in Equation 3.32.

$$ {}^{L}\boldsymbol{\gamma} = 9.8 \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}^{L} \frac{meters}{second^2} \tag{3.32} $$

Since acceleration is constant, this gravity model is not actually dependent on the object location, ${}^{L}\boldsymbol{r}_{B/L}$, as shown in Equation 3.30, but the notation of the more general gravity model is retained for the purpose of future development. Finally, the accelerometer measurement is described by Equation 3.31. Similar to the angular rate model, the measurement includes white noise, bias offset, scaling, and cross-coupling errors. The model given by the previous equations is included with the gyroscope model in the inertial sensor models block in Figure 3.3. However, this model must be similarly inverted to solve for the input to Equation 3.18. This inverse is represented by the inverse sensor model block in Figure 3.4 and is defined by the equations in Equation 3.33, Equation 3.34, and Equation 3.35.

$$ {}^{A}\hat{\boldsymbol{f}}_{A/L} = (\boldsymbol{I} + \boldsymbol{M}_A)^{-1}[{}^{A}\widetilde{\boldsymbol{f}}_{A/L} - {}^{A}\hat{\overline{\boldsymbol{f}}}_{A/L}] \tag{3.33} $$

$$ {}^{L}\hat{\boldsymbol{a}}_{A/L} = {}^{B}_{L}\hat{\boldsymbol{T}}^{\mathsf{T}}\,{}^{B}_{A}\boldsymbol{T}\,{}^{A}\hat{\boldsymbol{f}}_{A/L} + {}^{L}\boldsymbol{\gamma}({}^{L}\hat{\boldsymbol{r}}_{B/L}) \tag{3.34} $$

$$ {}^{L}\hat{\boldsymbol{a}}_{B/L} = {}^{L}\hat{\boldsymbol{a}}_{A/L} - {}^{B}_{L}\hat{\boldsymbol{T}}^{\mathsf{T}}[{}^{B}\hat{\dot{\boldsymbol{\omega}}}_{B/L} \times {}^{B}\boldsymbol{r}_{A/B} + {}^{B}\hat{\boldsymbol{\omega}}_{B/L} \times ({}^{B}\hat{\boldsymbol{\omega}}_{B/L} \times {}^{B}\boldsymbol{r}_{A/B})] \tag{3.35} $$

In light of the steps performed for the gyroscope model, the inverted accelerometer model is a relatively straightforward result of solving for the required terms. One significant issue with this model is the large degree of coupling between states. This coupling is

somewhat alleviated by moving the body frame origin to the accelerometer frame as previously mentioned. If this technique is not employed, an estimate of the angular acceleration, ${}^{B}\hat{\boldsymbol{\omega}}_{B/L}$, is required. If only angular rate measurements are available, the angular acceleration must be acquired through some sort of discrete derivative of these measurements. Care must be exercised in this case as methods of computing discrete derivatives tend to amplify noise in the input signal if not properly filtered. With the gyroscope and accelerometer models defined, the process model is completely specified.

3.4.3 Practical Considerations

There are several additional considerations related to the injection of inertial sensor measurements in these equations. The first issue is that of dynamic range. All inertial sensors have a finite dynamic range on their outputs. If this dynamic range is exceeded, due to shock or excessive vibration, the sensor outputs may become saturated. A robust filter must employ policies to detect and safely manage saturation. Techniques to accomplish detection and management are proposed by [68]. Another concern for system reliability is the inertial data stream integrity. The filter designed for this research assumes that the inertial sensors consistently generate measurements at the filter update rate defined by the interval T_s. A robust system must define fault detection and correction policies for dealing with lost or corrupted inertial data. Lastly, it is important to consider the validity of noise characteristics assigned to the stochastic inputs in these models. These models along with the UKF implementation assume all stochastic inputs are zero-mean, white, independent, and normally distributed. Real inertial sensors exhibit some amount of time correlation or even correlation among outputs as a result of environmental effects like temperature changes or electromagnetic interference. Depending on severity, these effects may be handled through the addition of fictitious noise or through additional modeling. In either case, the noise characteristics of the physical sensors must be characterized to properly tune the process model.

3.5 Filter Mechanization

The next component in this design is the navigation filter mechanization. This component consists of the mathematical definition of the filter processing steps using the process and measurement models as inputs. The UKF mechanization was previously described in Section 2.3.5, and the filter mechanization here follows closely from the previous definition. As before, the prediction and correction processes are implemented as independent operations in software to maximize generality. The prediction process is abstracted as the function in Equation 3.36, previously shown graphically in Figure 3.2 and Figure 3.3. This function accepts the process model function, $\boldsymbol{f}$, as an argument and performs the UKF prediction computation given by Equation 3.37. The scalar value N represents the number of states and the function $\boldsymbol{g}_{state}$ defines an arbitrary constraint on the final state. This constraint function is used to perform quaternion renormalization to enforce the unit norm constraint. In the future, this constraint should be applied before the covariance computation for greater stability, but (3.37) reflects the configuration applied in this research.

$$[\hat{\boldsymbol{x}}_k^-, \hat{\boldsymbol{P}}_k^-] = \boldsymbol{ukfPredict}(\hat{\boldsymbol{x}}_{k-1}^+, \hat{\boldsymbol{P}}_{k-1}^+, \boldsymbol{u}_{k-1}, \boldsymbol{Q}_{k-1}, \boldsymbol{f}, \boldsymbol{g}_{state}) \tag{3.36}$$

$$\begin{aligned}
\hat{\boldsymbol{x}}_{k-1}^{(i)} &= \hat{\boldsymbol{x}}_{k-1}^+ + \begin{cases} (N\hat{\boldsymbol{P}}_{k-1}^+)_i^{\mathsf{T}} & 1 \leq i \leq N \\ -(N\hat{\boldsymbol{P}}_{k-1}^+)_i^{\mathsf{T}} & N < i \leq 2N \end{cases} \\
\hat{\boldsymbol{x}}_k^{(i)-} &= \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}^{(i)}, \boldsymbol{u}_{k-1}) \\
\hat{\boldsymbol{x}}_{k,uc}^- &= \frac{1}{2N}\sum_{i=1}^{2N} \hat{\boldsymbol{x}}_k^{(i)-} \\
\hat{\boldsymbol{P}}_k^- &= \boldsymbol{Q}_{k-1} + \frac{1}{2N}\sum_{i=1}^{2N} (\hat{\boldsymbol{x}}_k^{(i)-} - \hat{\boldsymbol{x}}_{k,uc}^-)(\hat{\boldsymbol{x}}_k^{(i)-} - \hat{\boldsymbol{x}}_{k,uc}^-)^{\mathsf{T}} \\
\hat{\boldsymbol{x}}_k^- &= \boldsymbol{g}_{state}(\hat{\boldsymbol{x}}_{k,uc}^-)
\end{aligned} \tag{3.37}$$

For the initial filter implementation, all stochastic inputs are assumed to be additive and normally distributed, following the definition of the process model in Equation 2.18. These assumptions simplify sigma point selection by removing the extra sigma point scaling parameter mentioned in Section 2.3.5 and requiring no state augmentation to handle noise [6, 28]. The compromise for this simplicity is that inertial sensor noise, which is not purely additive and couples with attitude, must be effectively represented by an additive noise component. The additive noise is also expected to include discretization errors and any other unmodeled effects. As a result, some degree of filter tuning is required to balance stability and speed of convergence. A basic tuning methodology is detailed in [10].

As a note on implementation, the matrix square root operation in Equation 3.37 can be computed in several ways, exploiting the fact that $\hat{\boldsymbol{P}}_k^+$ is symmetric positive semidefinite. Empirical methods have shown the Cholesky decomposition to produce the best balance of numerical and computational performance among iterative and analytic methods [69]. For this reason, the Cholesky decomposition is employed in this design. In addition, no explicit state estimate covariance symmetrization or regularization is required since the UKF update equations naturally produce a symmetric positive semidefinite $\hat{\boldsymbol{P}}_k^-$ for real-valued states and symmetric positive semidefinite $\boldsymbol{Q}_k$. The fictitious process noise is assumed tuned to maintain a symmetric positive definite $\hat{\boldsymbol{P}}_k^-$ since the standard Cholesky decomposition is not possible for positive semidefinite matrices. This requirement is not essential, however, since extensions to the Cholesky algorithm exist for positive semidefinite matrices. Note that this form of the UKF also propagates the full state estimate covariance. Propagating the square root of the state estimate covariance is widely known to produce a far more stable and robust filter [26, 6]. The modular design of this framework allows for simple replacement of the current filter with the more stable square root UKF as a future enhancement.

The correction process is similarly defined as a high-level function in Equation 3.38 and is depicted by the filter correction block in Figure 3.3. This function includes the com-

putations defined in Equation 3.39 where $\boldsymbol{h}$ is the measurement model function. Similar to prediction, this correction function uses the state constraint function $\boldsymbol{g}_{state}$ and an arbitrary measurement constraint function $\boldsymbol{g}_{meas}$ to apply quaternion renormalization, where applicable. As in (3.37), the measurement constraint is likely more appropriately applied before the covariance and cross-correlation computations, but (3.39) reflects the implementation in this thesis.

$$[\hat{\boldsymbol{x}}_k^+, \hat{\boldsymbol{P}}_k^+] = \boldsymbol{ukfCorrect}(\hat{\boldsymbol{x}}_k^-, \hat{\boldsymbol{P}}_k^-, \widetilde{\boldsymbol{y}}_k, \boldsymbol{R}_k, \boldsymbol{h}, \boldsymbol{g}_{state}, \boldsymbol{g}_{meas}) \tag{3.38}$$

$$\begin{aligned}
\hat{\boldsymbol{x}}_k^{(i)} &= \hat{\boldsymbol{x}}_k^- + \begin{cases} (N\hat{\boldsymbol{P}}_k^-)_i^{\mathsf{T}} & 1 \le i \le N \\ -(N\hat{\boldsymbol{P}}_k^-)_i^{\mathsf{T}} & N < i \le 2N \end{cases} \\
\hat{\boldsymbol{y}}_k^{(i)} &= \boldsymbol{h}(\hat{\boldsymbol{x}}_k^{(i)}) \\
\hat{\boldsymbol{y}}_{k,uc} &= \frac{1}{2N}\sum_{i=1}^{2N} \hat{\boldsymbol{y}}_k^{(i)} \\
\hat{\boldsymbol{y}}_k &= \boldsymbol{g}_{meas}(\hat{\boldsymbol{y}}_{k,uc}) \\
\boldsymbol{P}_y &= \boldsymbol{R}_k + \frac{1}{2N}\sum_{i=1}^{2N}(\hat{\boldsymbol{y}}_k^{(i)} - \hat{\boldsymbol{y}}_{k,uc})(\hat{\boldsymbol{y}}_k^{(i)} - \hat{\boldsymbol{y}}_{k,uc})^{\mathsf{T}} \\
\boldsymbol{P}_{xy} &= \frac{1}{2N}\sum_{i=1}^{2N}(\hat{\boldsymbol{x}}_k^{(i)} - \hat{\boldsymbol{x}}_k^-)(\hat{\boldsymbol{y}}_k^{(i)} - \hat{\boldsymbol{y}}_{k,uc})^{\mathsf{T}} \\
\boldsymbol{K}_k &= \boldsymbol{P}_{xy}\boldsymbol{P}_y^{-1} \\
\hat{\boldsymbol{x}}_{k,uc}^+ &= \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\widetilde{\boldsymbol{y}}_k - \hat{\boldsymbol{y}}_k) \\
\hat{\boldsymbol{P}}_k^+ &= \hat{\boldsymbol{P}}_k^- - \boldsymbol{K}_k\boldsymbol{P}_y\boldsymbol{K}_k^{\mathsf{T}} \\
\hat{\boldsymbol{x}}_k^+ &= \boldsymbol{g}_{state}(\hat{\boldsymbol{x}}_{k,uc}^+)
\end{aligned} \tag{3.39}$$

The correction equations incorporate the measurement, $\widetilde{\boldsymbol{y}}_k$, and measurement noise covariance, $\boldsymbol{R}_k$. In a similar fashion, the correction equations allow for $\hat{\boldsymbol{P}}_k^+$ to remain sym-

metric positive definite for real-valued $\widetilde{\boldsymbol{y}}_k$, symmetric positive semidefinite $\boldsymbol{R}_k$, and properly tuned $\boldsymbol{Q}_k$. These two functions comprise the entire modularized filter mechanization.

In the present form, this design does not define a method for handling delayed measurement updates. Delayed measurement updates are necessary when sensor data arrive late or out of sequence [6]. In this case, the data must be applied retroactively to the filter, typically through some sort of rollback mechanism. The simplest form of this operation is to buffer measurements up to the maximum amount of expected lag and reprocess the existing and delayed measurements from the required point in time. This method clearly incurs a large computational cost. To avoid this performance penalty, it is possible to use a process known as retrodiction to predict the past state of the filter at the time the measurement was relevant. This process requires no excessive buffering and effectively performs reprocessing in a smaller number of steps. Other approaches also exist that make simplifying assumptions or approximations to achieve varying degrees of computational complexity [19]. Given their nature, both GNSS and LiDAR are likely to produce measurements with a significant delay between the time and availability of the measurement. For GNSS, this delay can be on the order of 100 milliseconds. Depending on the technique and complexity of LiDAR-aiding, this delay could potentially be much longer. This version of the filter makes allowances for a delayed update in the framework but does not actually implement any particular method.

3.6 Aiding-Sensor Models

As previously discussed, each aiding sensor is responsible for maintaining its own model defining the expected measurement as a function of the current state estimate. Each sensor must also provide its measurement noise covariance, $\boldsymbol{R}_k$. Using these two quantities, the filter can compute the measurement update gain, $\boldsymbol{K}_k$, to be applied. For the initial design, a crude position and velocity fix model is employed to simulate GNSS measurements. Unlike the inertial sensor models, this model does not include any installation offsets and simply

provides noisy position and velocity measurements. It is assumed that these measurement updates are acquired at a much lower rate than the inertial sensors. The representative measurement model is shown in Equation 3.40.

$$\widetilde{\boldsymbol{y}}_{fix,k} = \begin{bmatrix} {}^{L}\widetilde{\boldsymbol{r}}_{B/L} \\ {}^{L}\widetilde{\boldsymbol{v}}_{B/L} \end{bmatrix} = \begin{bmatrix} {}^{L}\boldsymbol{r}_{B/L} + {}^{L}\boldsymbol{v}_{\widetilde{\boldsymbol{r}}} \\ {}^{L}\boldsymbol{v}_{B/L} + {}^{L}\boldsymbol{v}_{\widetilde{\boldsymbol{v}}} \end{bmatrix} \tag{3.40}$$

$$\boldsymbol{R}_{fix,k} = \begin{bmatrix} \boldsymbol{\sigma}^2_{\widetilde{\boldsymbol{r}},fix} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\sigma}^2_{\widetilde{\boldsymbol{v}},fix} \end{bmatrix} \tag{3.41}$$

This component is represented by the position and velocity fix model implementation block in Figure 3.3. The position and velocity fix additive noise components, ${}^{L}\boldsymbol{v}_{\widetilde{\boldsymbol{r}}}$ and ${}^{L}\boldsymbol{v}_{\widetilde{\boldsymbol{v}}}$, are zero-mean and white with variances of $\boldsymbol{\sigma}^2_{\widetilde{\boldsymbol{r}},fix}$ and $\boldsymbol{\sigma}^2_{\widetilde{\boldsymbol{v}},fix}$, respectively. The measurement noise covariance, $\boldsymbol{R}_k$, incorporates these terms in a straightforward manner as shown in Equation 3.41. These components are also assumed uncorrelated with one another for this model, although in reality a true GNSS receiver would produce correlated measurements, since position and velocity are derived from common pseudorange measurements [10]. This model also lacks any representation of the GNSS receiver clock offset. Depending on the implemented level of GNSS coupling, it may be necessary to estimate the GNSS receiver clock offset as one of the filter states. No immediate LiDAR-aiding model is included in this design, as this simple position fix update is representative of position aiding through point cloud correspondence matching. Since LiDAR aiding in this manner may also produce estimates of rotational offsets, a more representative model would include an attitude fix as well. However, this simple model is sufficient to demonstrate the modularized filter.

3.7 Alignment and Initialization

Navigation system alignment refers to the process of determining the initial system attitude [70]. This process may also include establishing the starting position and velocity, but these states are often much more readily available through GNSS. Attitude poses a greater challenge since GNSS cannot measure it directly, and direct measurements, through magnetometers or a digital compass, often contain large errors. One of the most successful techniques for terrestrial applications is gyrocompassing, which attempts to determine attitude by measuring the acceleration of gravity and the angular rate of the earth's rotation while stationary. These measurements effectively provide two unit vectors which, apart from measurement errors, uniquely define the attitude with respect to the rotating earth (ECEF). Depending on the required accuracy, the alignment algorithm may even require a separate KF to fine tune the initial attitude [71]. Although this process is usually required, the filter implemented in this research employs a simplified approach. System initialization is performed using a standard measurement update of the position, velocity, and attitude states, as shown in Figure 3.3. This approach allows for bias estimation to occur immediately on system startup. To mitigate instability caused by invalid startup values, the state estimates are simply replaced by the measured states if the difference is above some threshold. This technique has proven useful to gently move the state estimate during GNSS measurement updates under similar conditions [19]. Unfortunately, direct measurement of the inertial sensor bias states is not possible, resulting in some degree of initial instability in these states. This behavior is only allowed in the initialization phase of the filter. The simulation architecture in Figure 3.3 includes logic to switch the applied measurement update from initialization mode to the position/velocity fix mode after this phase is complete.

The initialization measurement update model, shown in Equation 3.42, closely resembles that of the position and velocity fix update.

$$\widetilde{\boldsymbol{y}}_{init,k} = \begin{bmatrix} {}^{L}\widetilde{\boldsymbol{r}}_{B/L} \\ {}^{L}\widetilde{\boldsymbol{v}}_{B/L} \\ {}^{B}_{L}\widetilde{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} {}^{L}\boldsymbol{r}_{B/L} + {}^{L}\boldsymbol{v}_{\widetilde{\boldsymbol{r}}} \\ {}^{L}\boldsymbol{v}_{B/L} + {}^{L}\boldsymbol{v}_{\widetilde{\boldsymbol{v}}} \\ {}^{B}_{L}\boldsymbol{q} + {}^{L}\boldsymbol{v}_{\widetilde{\boldsymbol{q}}} \end{bmatrix} \tag{3.42}$$

$$\boldsymbol{R}_{init,k} = \begin{bmatrix} \boldsymbol{\sigma}^2_{\widetilde{\boldsymbol{r}},init} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\sigma}^2_{\widetilde{\boldsymbol{v}},init} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\sigma}^2_{\widetilde{\boldsymbol{q}},init} \end{bmatrix} \tag{3.43}$$

This approach to initialization essentially defers the alignment problem in a generalized manner since external state measurements are required. For this research, initialization measurements are acquired by adding zero-mean, white, uncorrelated noise to the true object position, velocity, and attitude. The measurement noise covariance, $\boldsymbol{R}_{init,k}$, given by Equation 3.43 reflects the statistics of this noise. Unlike standard gyrocompassing, this method allows for initialization while the object is already in motion using the standard measurement update, assuming another source of measurements is available. With the initialization procedure defined, the technical approach to this design is complete. The next chapter defines the simulation configuration and analyzes simulation results.

CHAPTER 4

SIMULATION AND RESULTS

This section describes the simulation used to verify the functionality of the filter framework design. This design includes the process, inertial sensor, and measurement models discussed in Chapter 3. The initial filter implementation and simulation is written completely in MATLAB. Verification occurs through two performance tests. The first series of tests checks system performance and tuning using Monte Carlo trajectory simulation. For this test, random trajectories and simulated sensor data are generated by a stochastic algorithm. The navigation filter is then used to estimate each trajectory from the sensor data in order to check for reliability and ensure consistent performance.

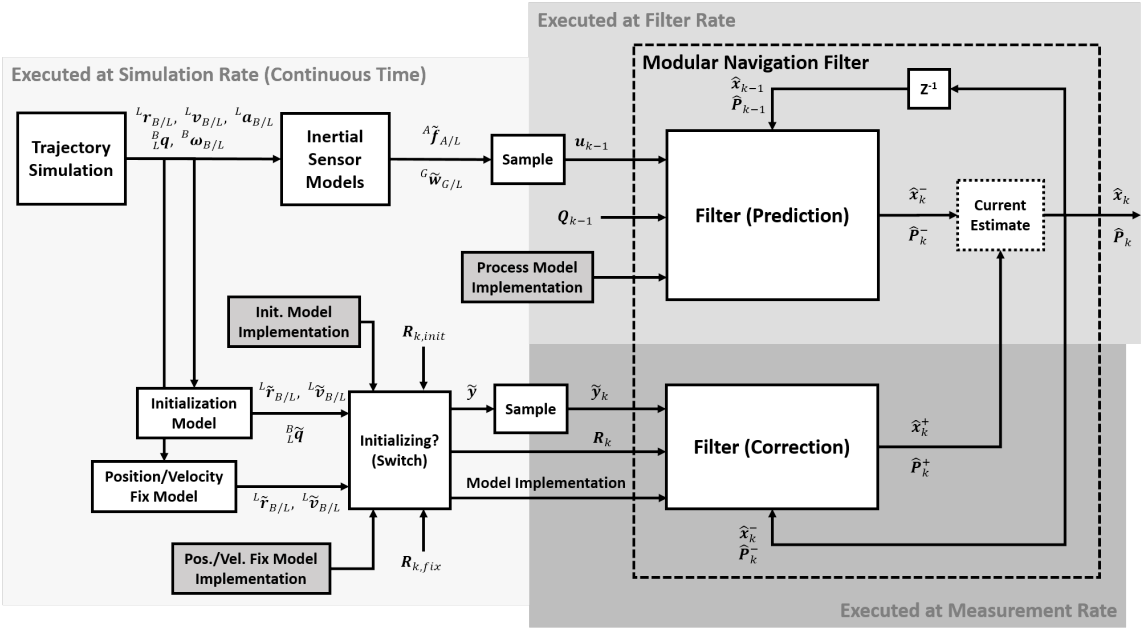

Figure 4.1: Simulation Architecture

The second test evaluates system performance using more realistic trajectory data from the X-Plane flight simulator. The purpose of this second test is to ensure system performance in a more representative environment. Within the context of the simulation architecture, as shown in Figure 4.1, these two tests only differ in the implementation of the trajectory simulation block. This chapter first describes the system configuration used for testing, and then proceeds with descriptions of the test processes and their respective results.

4.1 System Configuration

The configuration for this navigation system has many parameters. Values selected for performance testing are determined through requirements for internal research projects at the Electro-Optical Systems Laboratory within GTRI. This section contains tabulation of all required parameters for the filter. The specified variances or variables shown in parentheses in each the table are derived from the corresponding values. Simulation rate parameters are given in Table 4.1. These parameters define the rates of simulated flight data, navigation filter updates, and aiding sensor measurements. The simulated flight data rate is set higher than the filter and sensor data rates in order to simulate continuous time. The filter rate is determined by the inertial sensor data rate as well as the overall desired data rate for the system. A navigation filter rate of 200 Hz is necessary in order to meet the requirements of the research effort. The aiding sensor data rate is selected to represent GNSS measurements, as many GNSS receivers produce measurements at a rate of 1 to 5 Hz [10].

Table 4.1: Simulation Sampling Rates and Frequencies

Simulation ($T_{s,sim}$)	0.0025 sec	400.0 Hz
Filter ($T_{s,filt}$)	0.005 sec	200.0 Hz
Position/Velocity Fix ($T_{s,fix}$)	1.0 sec	1.0 Hz

Table 4.2: Inertial Sensor Parameters

IMU	
Bandwidth	100.0 Hz
Data Rate ($T_{s,filt}$)	200.0 Hz
Gyroscope	
Bias Instability ($\boldsymbol{\sigma}^2_{\dot{\tilde{\boldsymbol{\omega}}}}$)	1.0 deg/hour
Angle Random Walk ($\boldsymbol{\sigma}^2_{\boldsymbol{w}_G}$)	0.02 deg/$\sqrt{}$hour
Scale Factor Error ($\boldsymbol{M}_G$)	2.0×10^{-4}
Accelerometer	
Bias Instability ($\boldsymbol{\sigma}^2_{\dot{\tilde{\boldsymbol{f}}}}$)	100.0 μg
Velocity Random Walk ($\boldsymbol{\sigma}^2_{\boldsymbol{w}_A}$)	100.0 μg/$\sqrt{}$Hz
Scale Factor Error ($\boldsymbol{M}_A$)	2.0×10^{-4}

Parameters for the inertial sensor models reflect the performance specifications for the Systron Donner Inertial SDI500-AE00 MEMS Quartz Tactical IMU. This IMU was selected as a representative of the state of the art in MEMS IMU technology. Parameters in Table 4.2 are derived from the SDI500 specifications [72]. The inertial sensor parameters are used by the forward sensor models to produce simulated measurements. These parameters also inform tuning of the process noise covariance, $\boldsymbol{Q}_k$, but the final process noise is manually tuned to account for integration and numerical errors as well.

Table 4.3: Position/Velocity Fix Parameters

Position and Velocity Fix Standard Deviations	
Position ($\boldsymbol{\sigma}_{\tilde{\boldsymbol{r}},fix}$)	1.0 m
Velocity ($\boldsymbol{\sigma}_{\tilde{\boldsymbol{v}},fix}$)	0.1 m/s

The position and velocity fix measurements are generated from additive zero-mean, white, normally-distributed noise. Corresponding standard deviations are given in Table 4.3. The position/velocity fix standard deviations are derived from a combination

of EOSL research requirements and a survey of state-of-the-art GNSS receiver capabilities [73, 74]. These performance values are assumed achievable with a high-performance GNSS receiver.

Table 4.4: Initialization Parameters

General Parameters	
Initialization Time	60 sec
Initialization Measurement Standard Deviations	
Position ($\boldsymbol{\sigma}_{\widetilde{\boldsymbol{r}},init}$)	1.0 m
Velocity ($\boldsymbol{\sigma}_{\widetilde{\boldsymbol{v}},init}$)	0.1 m/s
Quaternion Attitude ($\boldsymbol{\sigma}_{\widetilde{\boldsymbol{q}},init}$)	5.0×10^{-5}

The next set of parameters, shown in Table 4.4, pertain to the states provided to the initialization measurement update. This update is performed at the navigation filter rate using measurements output from an external process, or presumably, an alignment algorithm. These parameters consist of standard deviations for the initialization measurements.

The values for process noise parameters are selected by manual tuning. These values ensure adequate performance over all test cases to verify filter operation while maintaining stability. However, use in a real-world application would require a more rigorous tuning process. The values determined by manual tuning are shown in Table 4.5.

Table 4.5: Process Noise

Process Noise Standard Deviations	
Position ($\boldsymbol{\sigma}_{\boldsymbol{r}}$)	1.0000×10^{-3} m
Velocity ($\boldsymbol{\sigma}_{\boldsymbol{v}}$)	1.3909×10^{-2} m/s
Accelerometer Bias ($\boldsymbol{\sigma}_{\overline{\boldsymbol{f}}}$)	9.8105×10^{-4} m/s^2
Quaternion Attitude ($\boldsymbol{\sigma}_{\boldsymbol{q}}$)	1.0000×10^{-3}
Gyroscope Bias ($\boldsymbol{\sigma}_{\overline{\boldsymbol{\omega}}}$)	4.8481×10^{-6} deg/sec

4.2 Filter Performance Evaluation

Due to the intended application of this research, no quantitative performance criteria are imposed on the system. The purpose of testing in this case is to ensure correct filter and initial model implementation. The application models are intended to change in the future depending on research goals at EOSL. In lieu of the lack of quantitative requirements, the filter is evaluated with respect to several desired qualitative performance criteria. The following sections describe these criteria and the results expected when each is properly met.

4.2.1 Filter Stability

This research utilizes a less rigorous practical definition of filter stability. In a qualitative sense, filter stability refers to convergence of the state estimate to the true state value over time [10, 6]. The key concern for this research is that the filter output does not experience unbounded growth or undamped oscillations typical of marginal stability. This notion of stability is dependent on correct filter and model implementation as well as adequate tuning of the process noise covariance. Checking for a stable filter amounts to verifying that the filter output does not diverge or oscillate excessively.

4.2.2 Accurate Covariance Estimation

One of the most useful properties of the KF, and other similar filters, is the ability to estimate the covariance of the state estimate error. The state estimate error is defined as the difference between the true and estimated states. The covariance estimate is accurate when the process and measurement models meet all the prerequisite assumptions for the filter and perfectly capture all of the system dynamics. Any deviation from these assumptions adversely affects the covariance estimate as well as the state estimate itself. The degree of model incompatibility can be measured from the statistics of the state estimate during

simulation. This measurement involves the comparison of the state estimate error to an envelope defined by the estimate error variance. To perform a qualitative comparison, an envelope of twice the standard deviation (two-sigma) is plotted with each state estimate error as a visual reference. The two-sigma envelope of an accurate state estimate error covariance bounds the state estimate error 95 percent of the time. This condition is easily verified through inspection of the estimate error results of the simulation.

4.2.3 Uncorrelated Measurement Residual

An important result of the KF derivation is that the measurement residual is zero-mean and uncorrelated in time during correct operation [6]. The measurement residual, also called the innovation, is the difference between the actual and the *a priori* measurement prediction based on $\hat{\boldsymbol{x}}_k^-$. This condition is verified by computing the innovation at each time step and checking for reasonably uncorrelated behavior [75]. The required analysis is performed in post-processing by computing the autocorrelation and mean of the innovation. The important benefit of this test is that, unlike the first two tests, the innovation can be computed even when the true system state is hidden since it only relies on the actual and predicted measurements. This property allows for application of the measurement residual test during real-time filter operation. Testing of these three criteria is performed on the outputs of the random walk and X-Plane trajectory simulations as required. The specific purpose and methodology of these simulations are discussed in the following sections.

4.3 Random Walk Trajectory Simulation

The first series of tests are performed using Monte Carlo trajectory simulation. The purpose of these tests is to check filter performance over a large number of randomly generated trajectories and to initially tune the process noise. To accomplish these goals, a simple

random walk trajectory model is employed. This trajectory model is given by the equations in Equation 4.1.

$$
\begin{aligned}
{}^{L}\dot{\boldsymbol{r}}_{B/L} &= {}^{L}\boldsymbol{v}_{B/L} \\
{}^{L}\dot{\boldsymbol{v}}_{B/L} &= {}^{L}\boldsymbol{a}_{B/L} \\
{}^{B}_{L}\dot{\boldsymbol{q}} &= \frac{1}{2}\boldsymbol{\Omega}({}^{B}\boldsymbol{\omega}_{B/L})\,{}^{B}_{L}\boldsymbol{q} \\
||\,{}^{B}_{L}\boldsymbol{q}||_2 &= 1 \\
{}^{L}\dot{\boldsymbol{a}}_{B/L} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{\sigma}_{\dot{\boldsymbol{a}}_{traj}}) &\quad {}^{B}\dot{\boldsymbol{\omega}}_{B/L} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{\sigma}_{\dot{\boldsymbol{\omega}}_{traj}})
\end{aligned}
\tag{4.1}
$$

In the trajectory model, normally distributed white noise is used to drive random walk of the vehicle acceleration (${}^{L}\boldsymbol{a}_{B/L}$) and angular rates (${}^{B}\boldsymbol{\omega}_{B/L}$). These quantities are input into the standard kinematic process model and integrated to produce the true trajectory using the second-order modified Heun integration scheme presented in Section 3.3.4. The standard deviations for the random walk process are given in Table 4.6.

Table 4.6: Trajectory Model Parameters

Random Walk Trajectory Standard Deviations	
Acceleration Random Walk ($\boldsymbol{\sigma}_{\dot{\boldsymbol{a}}_{traj}}$)	0.8 m/s^3
Angular Rate Random Walk ($\boldsymbol{\sigma}_{\dot{\boldsymbol{\omega}}_{traj}}$)	0.5 deg/sec^2

This approach was developed to generate crudely-representative acceleration and angular rate data. However, the result is that the model lacks the smooth sinusoidal and step behavior typical of pilot inputs. This difference is apparent by comparing the accelerations and angular rates from the Monte Carlo trajectory model and a test flight in X-Plane. Simulated accelerations and angular rates are shown in Figure 4.2 and Figure 4.3. Note that these quantities represent the true state values rather than simulated measurements, and do not include biases or noise generated from the inertial sensor models.

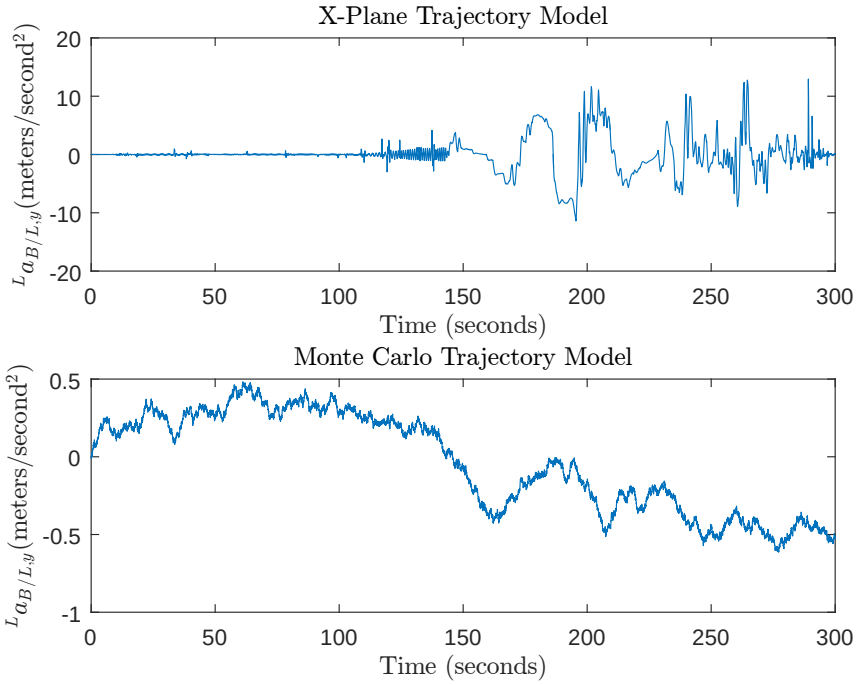

Figure 4.2: Sample Accelerations

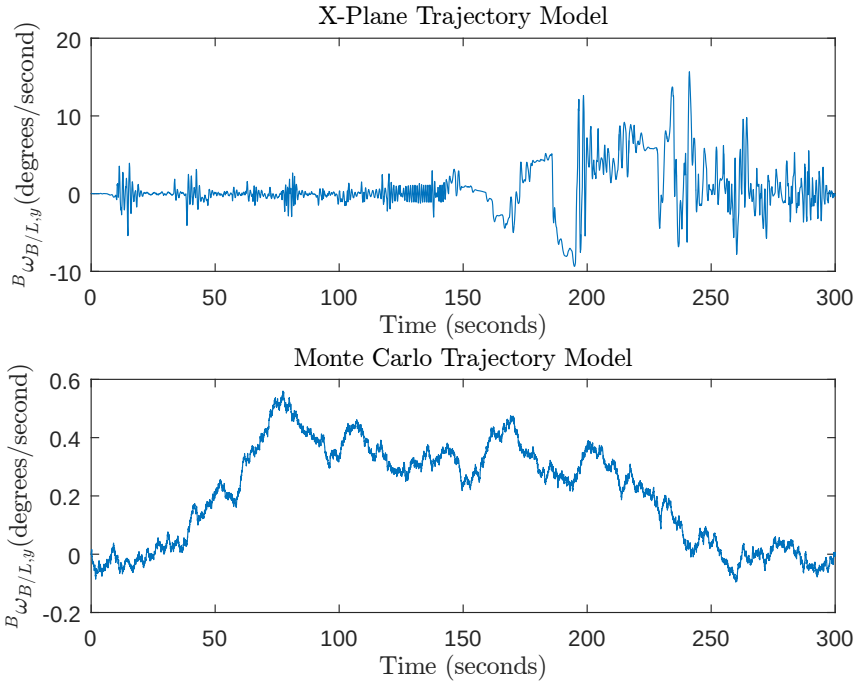

Figure 4.3: Sample Angular Rates

A number of differences are apparent from Figure 4.2 and Figure 4.3. As previously mentioned, the X-Plane trajectory contains far more structure due to pilot inputs. Regions that resemble noise in the X-Plane acceleration and angular rate inputs are actually higher-frequency sinusoidal components. The relative scale of the random trajectory inputs is also much smaller in order to limit the distance of the overall excursion to within the same relative range of the X-Plane simulation. Despite their overall smaller magnitude, the random trajectory inputs exhibit a significant constant component, whereas the X-Plane inputs appear centered around zero with relatively short transients when control inputs occur. These stark differences in behavior provide the motivation for X-Plane simulator testing. For a real-world system, accurate flight dynamics models representing the full range of operating environments and vehicles would be necessary to perform rigorous Monte Carlo testing. This random trajectory test serves as a simplified alternative for this research.

For this test, a series of fifty random trajectories are generated from the trajectory model. The filter processes simulated inertial measurements generated from the random trajectory data. Results are first presented for a single trial to provide an example of the data products from each test. This trial is analyzed for filter stability, accurate covariance estimation, and uncorrelated measurement residuals as described above. Next, comparisons are performed on the total root-mean squared (RMS) and final states errors across trials to ensure relatively consistent filter performance over all the trajectories.

4.3.1 Single Trial

The first few figures in this section show the standard results produced by a single trial of the filter. Each trial in this case tests filter performance on a new random trajectory. These results are intended to provide insight into the characteristic simulation outputs prior to higher-level comparison among trials. The true and estimated positions are shown in Figure 4.4. The position track demonstrates unnatural smoothness caused by the crude random trajectory model. The axis extents required to view the full track also obscure the actual

trajectory errors because the true and estimated trajectories are nearly perfectly coincident at this level of detail. These errors are more apparent by computing the difference of the true and estimated position states as shown in Figure 4.5. The estimate error computation requires resampling of simulation rate state data. Data at the simulation rate, $T_{s,sim}$, must be reduced to the filter sampling rate, $T_{s,filt}$. For rates that are not related by harmonics, downsampling is accomplished through linear interpolation. The position estimate error in Figure 4.5 includes an envelope representing the two-sigma bounds computed from the diagonal of the estimate error covariance.

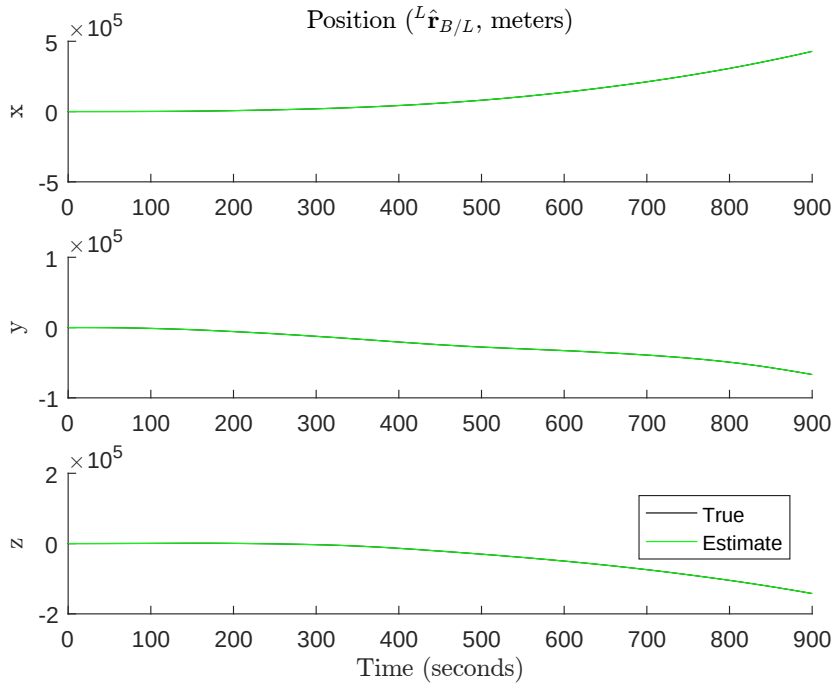

Figure 4.4: Random Trajectory - Position
(True and estimated values are coincident at this scale)

Based on the figure, the estimated variance progresses through three stages. At the start of the trial, the variance decreases rapidly from an arbitrarily large value. This value is set during filter startup to represent a complete lack of knowledge of the initial system state. A

startup value of 40 was arbitrarily assigned to each of the states at startup. In reality, this value is not actually large enough to represent a complete lack of information, but using appropriately large values risks harming matrix conditioning if states converge at different rates. These effects are mitigated in this research by gently moving state estimates from their startup values during initialization.

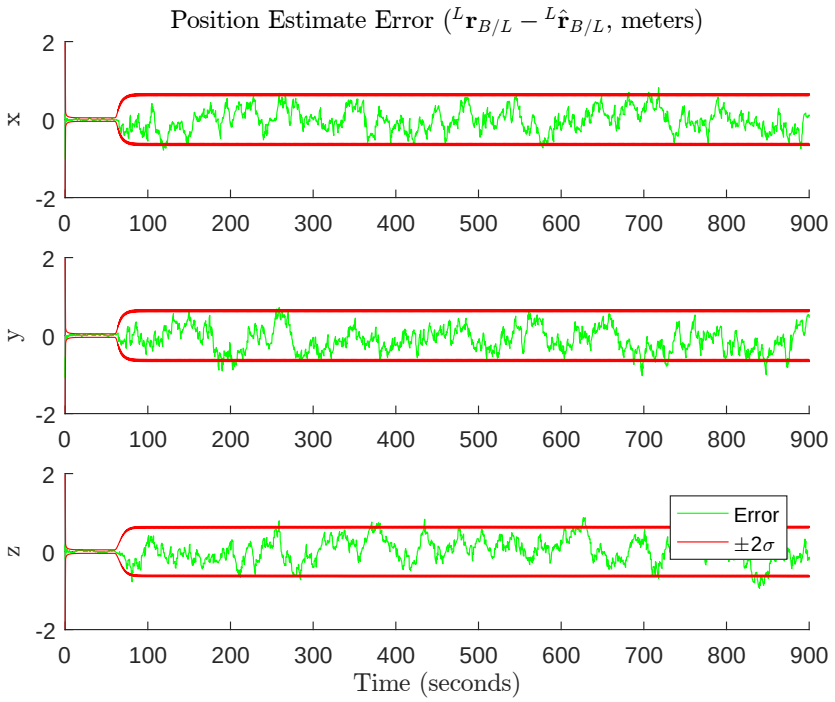

Figure 4.5: Random Trajectory - Position Estimate Error

After convergence, the estimated variance assumes a relatively constant value that reflects the uncertainty in initialization measurements. As previously mentioned, initialization is simulated through a measurement update, during which position, velocity, and attitude measurements are received at the filter rate of $T_{s,filt}$. Bias estimation occurs simultaneously with measurement updates through the state update. This phase continues until the initialization period completes. After initialization, the filter begins normal operation using simulated GNSS measurements. At the start of this phase, the estimated variance increases

to the value afforded by GNSS corrections. Subsequent lower-rate measurement updates cause the estimated variance to assume a sawtooth waveform, enlarged in Figure 4.6. In this waveform, local minima occur when GNSS corrections are received. The increase following each minimum is caused by the short interval of DR between measurement updates. The estimate error exhibits similar behavior with small discontinuities at each update and smooth prediction between updates. This behavior is characteristic of all filter states since each is affected by the measurement update. During all phases, the two-sigma envelope is expected to bound the estimate error with about 95 percent confidence.

The position estimate error in Figure 4.5 provides the required information to analyze the position estimate in terms of the first two qualitative criteria. Based on the estimate error behavior, the position estimate appears stable and bounded by the two-sigma envelope for about 95 percent of the estimation time. The position measurement residual, given in Figure 4.7, provides the information required for the last qualitative criterion.

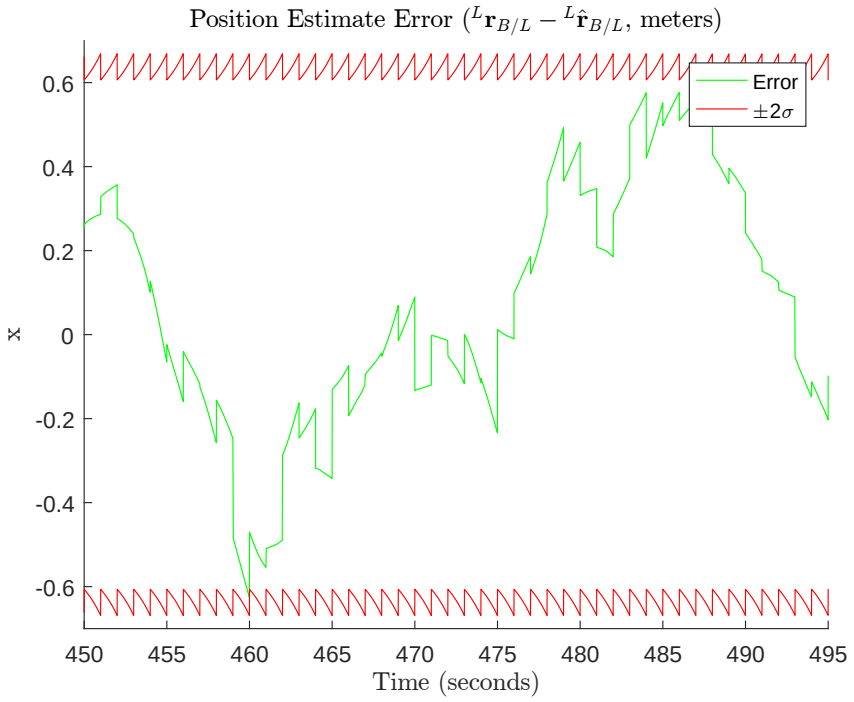

Figure 4.6: Random Trajectory - Position Estimate Error - Enlarged

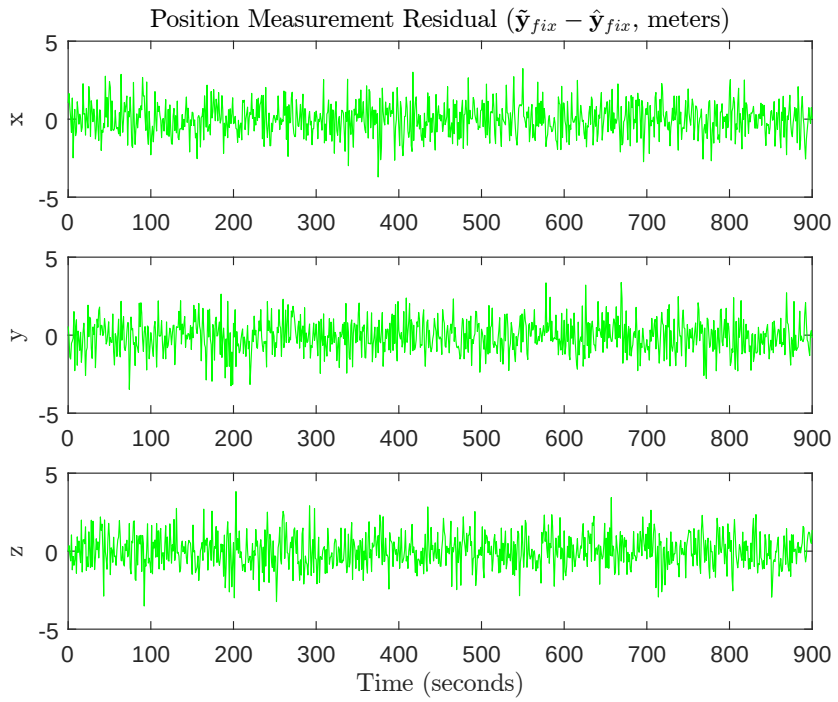

Figure 4.7: Random Trajectory - Position Measurement Residual

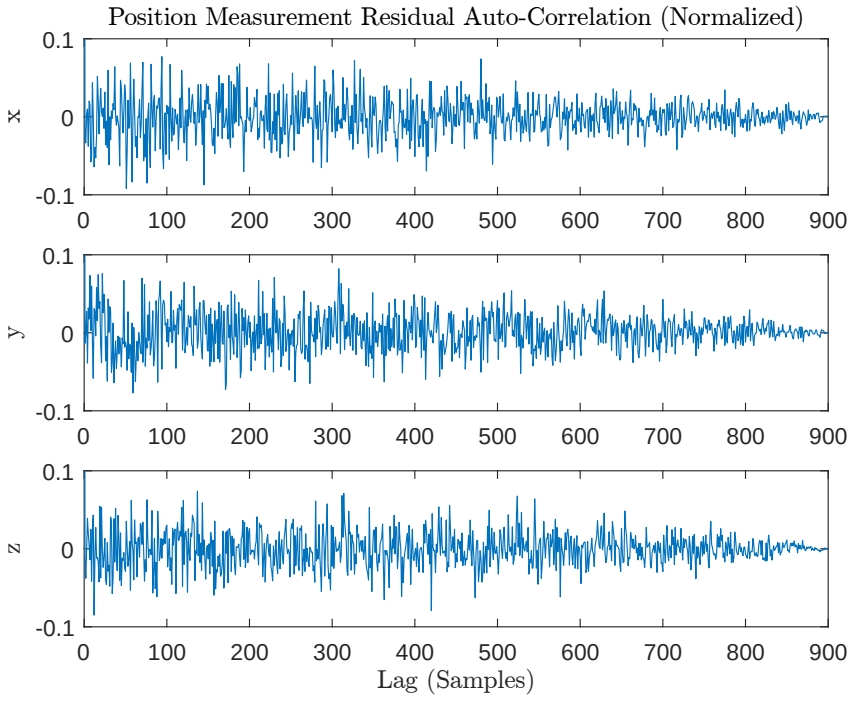

Figure 4.8: Random Trajectory - Position Measurement Residual Auto-Correlation

The last qualitative criteria requires that the position measurement residual is zero-mean and uncorrelated in time. The behavior in Figure 4.7 demonstrates roughly correct behavior of the mean and autocorrelation. The autocorrelation is further analyzed through numerical computation, producing Figure 4.8. This figure demonstrates relatively low correlation at non-zero lags which confirms adherence to the final qualitative criteria. Due to the axis extents, the correlation value of one at zero lag is not visible. Given the above analysis, the position estimate meets all three qualitative criteria. This same methodology is applied to the remaining filter states.

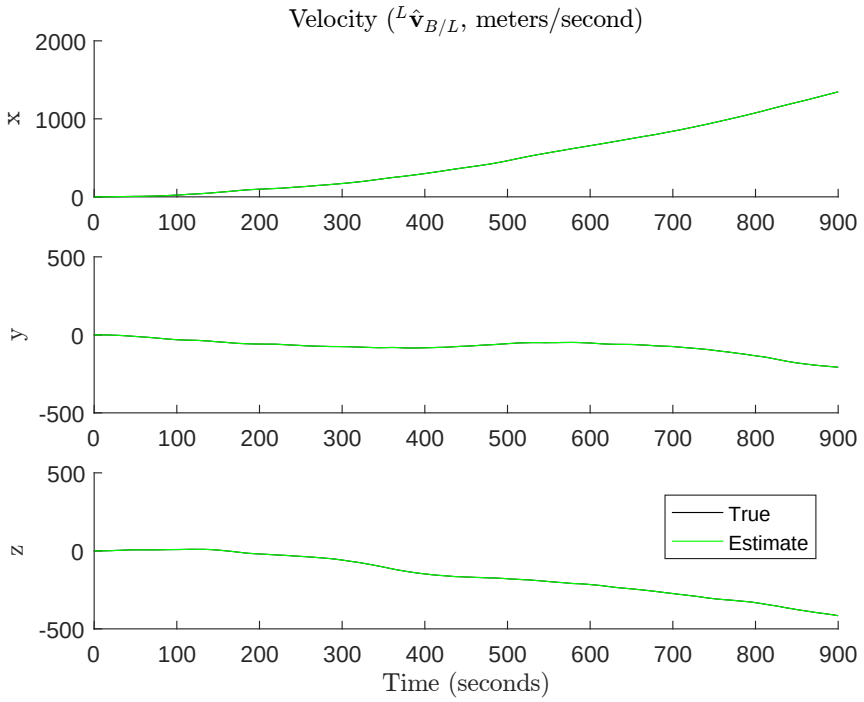

Figure 4.9: Random Trajectory - Velocity
(True and estimated values are coincident at this scale)

The state and estimate error plots for velocity are given in Figure 4.9 and Figure 4.10. Once again, the velocity estimate error and covariance estimate produce the required information. From these plots, the velocity estimate appears to maintain stability and remain

within the two-sigma bounds. There is one unusual feature of the velocity estimate error variance in Figure 4.10. The z-axis variance estimate is noticeably smaller than those of the other two axes. The exact cause of this difference is unknown, as all of the noise inputs for velocity are uniform. However, this issue is reconsidered in light of the accelerometer bias estimate error in Figure 4.17.

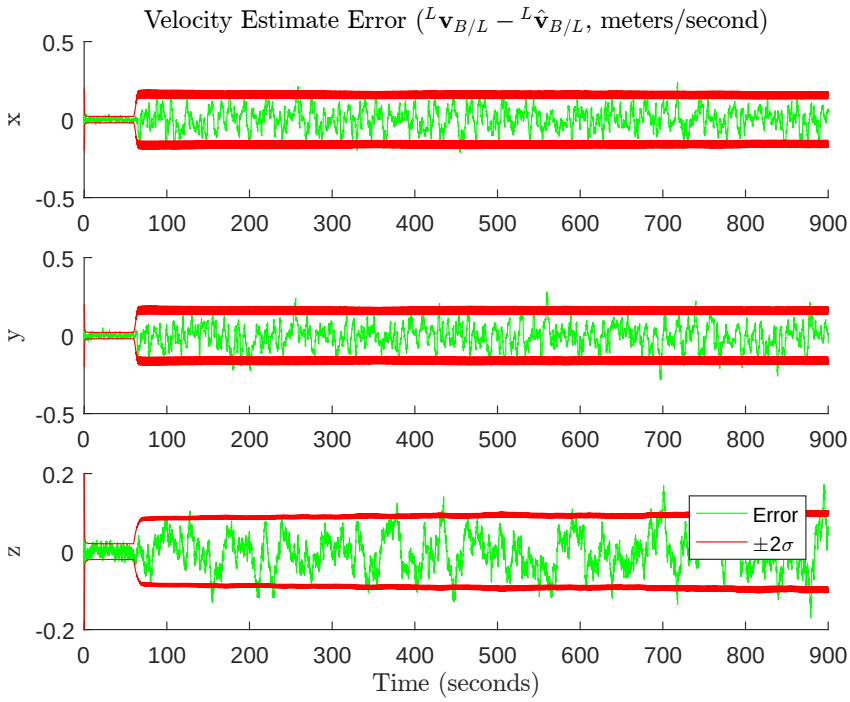

Figure 4.10: Random Trajectory - Velocity Estimate Error

The velocity measurement residual and correlation, in Figure 4.11 and Figure 4.12, again provide the information required for the third criterion. The velocity measurement residual is relatively uncorrelated, containing about the same amount of correlation as the position measurement residual. Based on these results, the velocity state estimate appears to meet all three criteria as well.

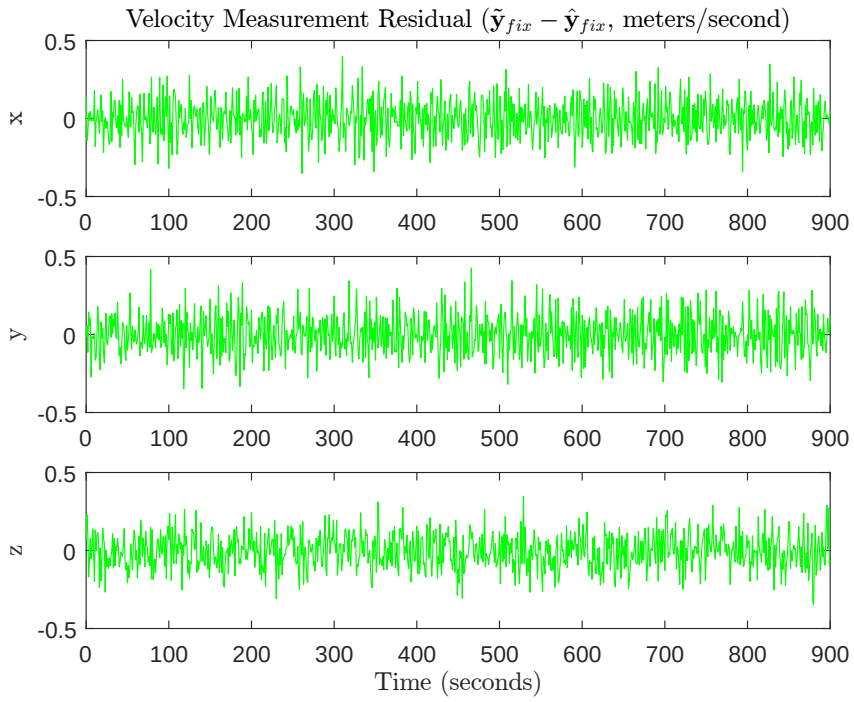

Figure 4.11: Random Trajectory - Velocity Measurement Residual

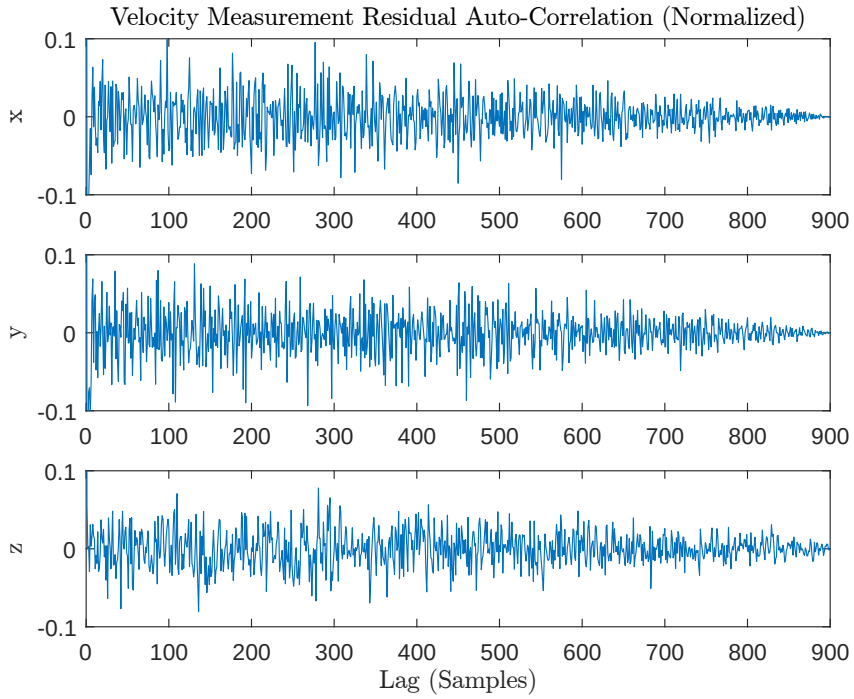

Figure 4.12: Random Trajectory - Velocity Measurement Residual Auto-Correlation

Direct analysis of the raw attitude state is not generally useful because the physical meaning of the quaternion parameters lacks a natural intuition. The quaternion attitude state is first converted to a Tait-Bryan angle representation for analysis. The true and estimated attitude states are given in Figure 4.13. This representation also possesses an apparent weakness due to wrapping of the angles. In this form, the Tait-Bryan angles are wrapped to ± 180 degrees in heading and roll. The pitch angle is constrained to ± 90 degrees. This form differs slightly from the standard definition of heading, which is usually constrained to the interval, $[0, 360)$, but conversions between forms are reasonably trivial.

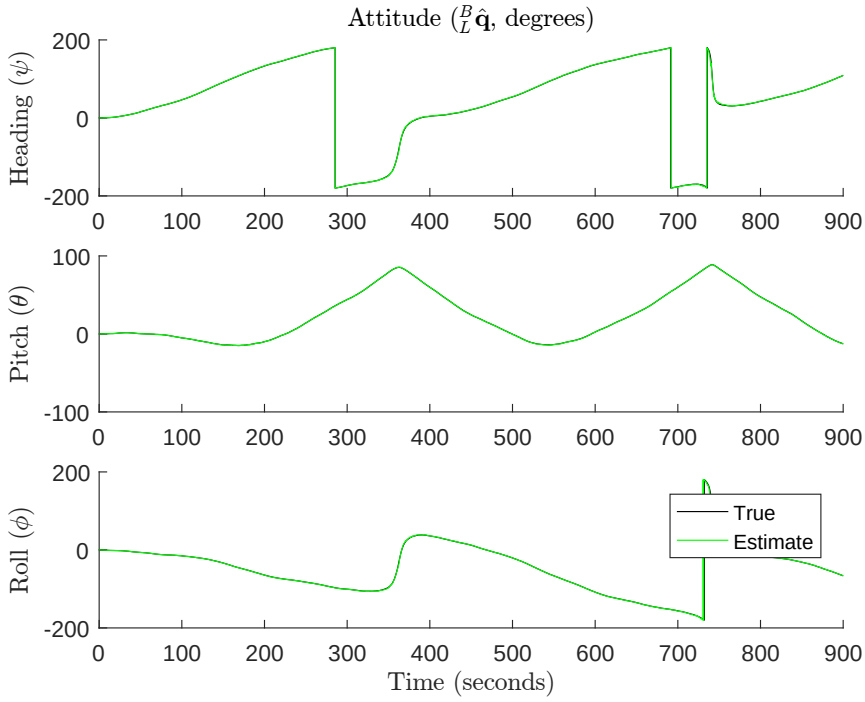

Figure 4.13: Random Trajectory - Attitude (Tait-Bryan Angles)
(True and estimated values are coincident at this scale)

As before, the attitude estimate error is a more useful quantity for evaluation of the qualitative criteria. However, analysis of the attitude estimate error requires greater care since rotational errors cannot be computed through subtraction of rotational parameters.

Alternative differencing methods exist for both the DCM and quaternion representations. This research applies the error computation defined through conjugate quaternion multiplication in [63]. The Tait-Bryan attitude errors shown in Figure 4.14 are computed using this method.

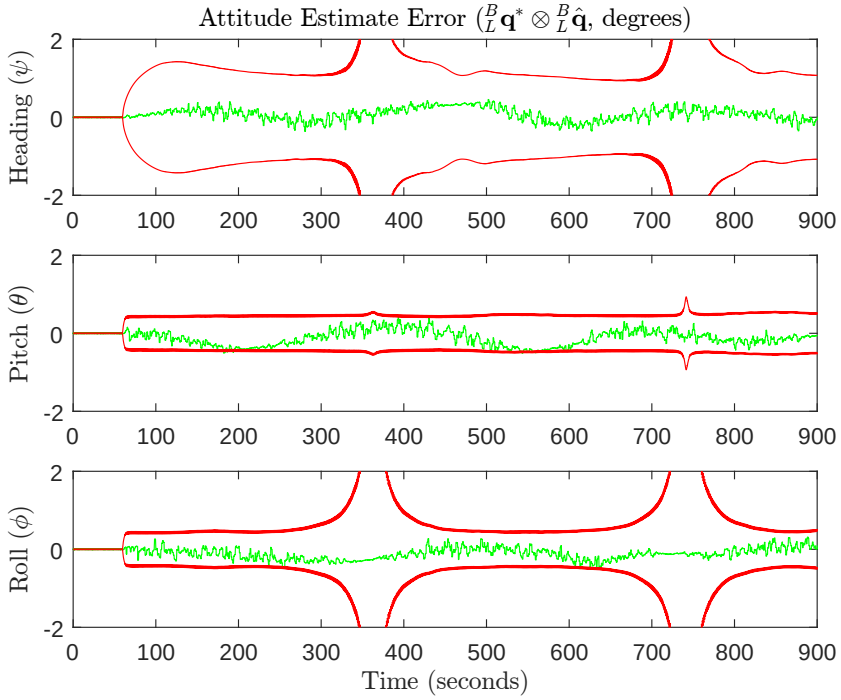

Figure 4.14: Random Trajectory - Attitude Estimate Error (Tait-Bryan Angles)

A special method of conversion is also required for the quaternion covariance in order to plot the two-sigma envelope. The method used to produce Figure 4.14 linearizes the quaternion to Tait-Bryan angle conversion in [58] and applies the standard linear covariance transform to approximate the resulting distribution [76]. Based on Figure 4.14, the attitude estimate is stable and does not exceed the two-sigma bounds defined by the transformed quaternion covariance. However, there are two sharp discontinuities that occur in the estimate variance itself. The locations of these discontinuities coincide with sharp changes in the attitude state in Figure 4.13, most noticeably in the pitch axis. Despite their appear-

ance, these seemingly discontinuous features represent a smooth trajectory. The unusual behavior in Figure 4.13 is the result of the Tait-Bryan representation itself. The quaternion covariance transformation is similarly influenced at these locations, causing the discontinuities in Figure 4.14. This conclusion is further supported by inspection of the quaternion attitude estimate errors and two-sigma bounds in Figure 4.15.

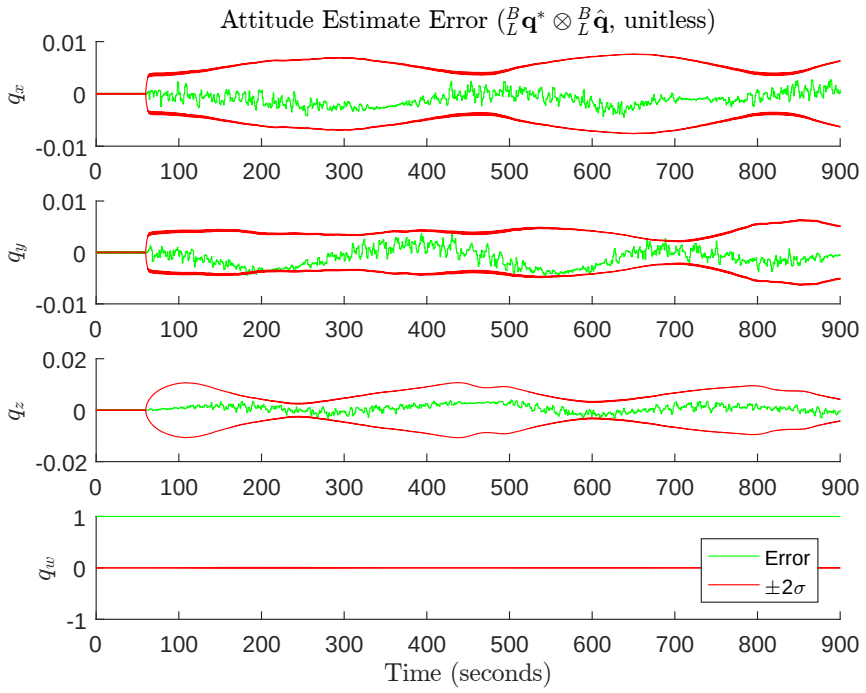

Figure 4.15: Random Trajectory - Attitude Estimate Error (Quaternion)

In Figure 4.15, the estimated quaternion attitude variance appears smooth, unlike that of Figure 4.14. The estimate error similarly remains within the two-sigma bounds on all but the q_w component. This irregularity is due to the fact that the covariance propagation in the filter does not account for the quaternion unit norm constraint, despite the renormalization performed on the state itself. However, the quaternion error reflects this unit norm constraint, which causes the scalar component, q_w, to remain close to unity for small angle errors. This effect serves as another justification for propagating a quaternion error state of

three parameters rather than the full quaternion attitude, since the q_w parameter does not add useful information. Based on the previous results, the attitude estimate satisfies the first two qualitative criteria. The third criteria does not apply in this case because no direct heading measurements are available, and therefore, no attitude measurement residual can be computed.

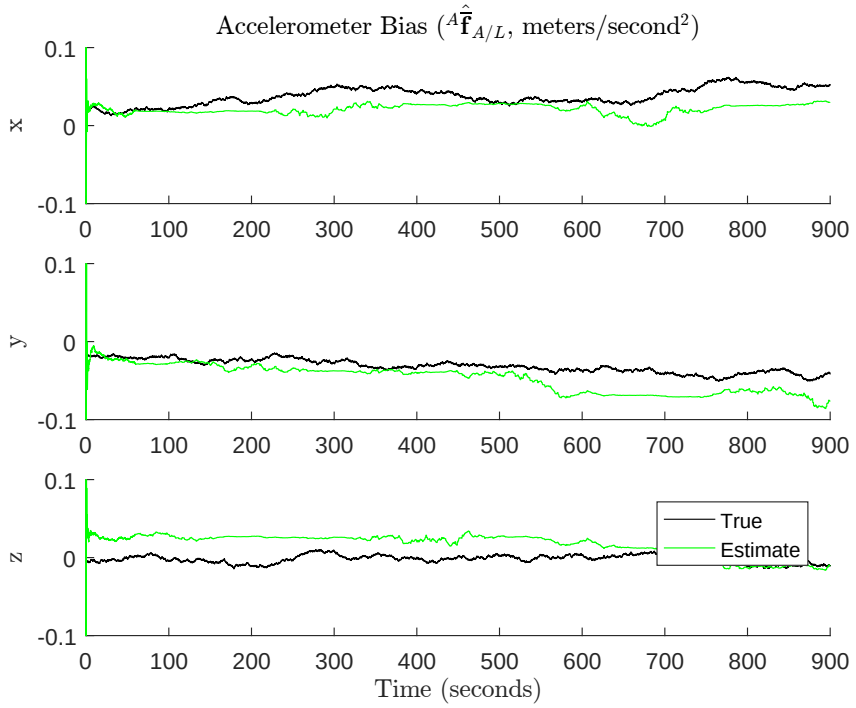

Figure 4.16: Random Trajectory - Accelerometer Bias

The accelerometer bias state and estimate error are given in Figure 4.16 and Figure 4.17. The random walk behavior of the true bias state is clearly visible in Figure 4.16. As expected, the estimate roughly tracks the changes in the true bias. The estimate signal is piecewise constant, reflecting the process model assumption that the bias is constant. The estimated bias exhibits some initial instability. This instability is caused by the method of initialization. The startup value of each state is gently moved to the measured value for initialization, but no direct measurements of the inertial sensor biases exist. The initializa-

tion model described in Section 3.7 assumes that only position, velocity, and attitude states are available for initialization, although a proper alignment algorithm would likely produce bias estimates as well. As a result, the startup value for the bias estimate is set to zero and differs from the true startup value. Combined with the arbitrarily high initial state variance, this initial condition causes brief instability and an offset, particularly in the z-axis bias potentially due to initial alignment with the acceleration of gravity. Due to coupling between bias states, this offset appears to affect the x and y axes as well. This effect is more noticeable considering the accelerometer bias estimate error in Figure 4.17.

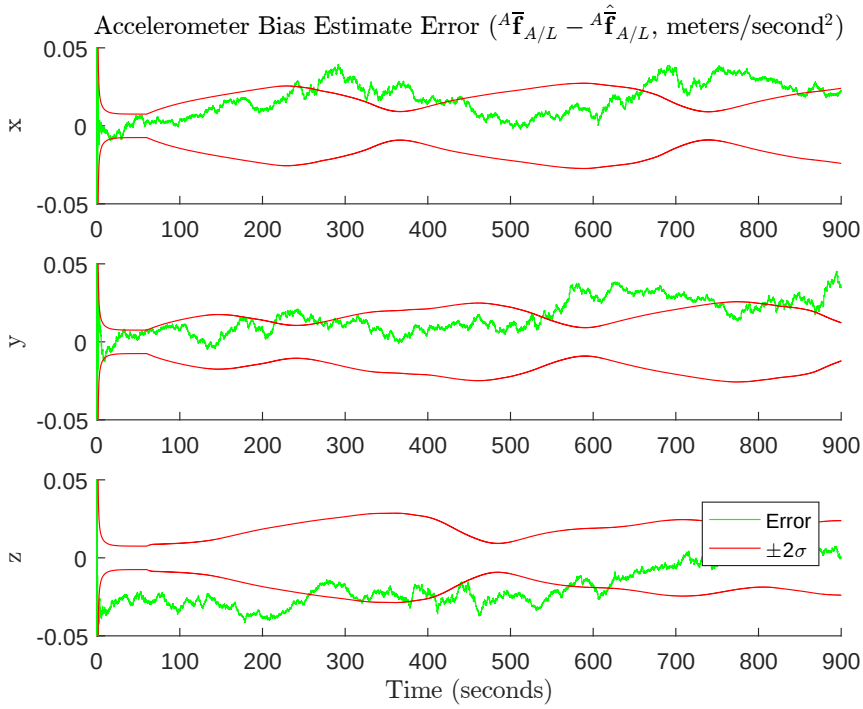

Figure 4.17: Random Trajectory - Accelerometer Bias Estimate Error

Despite the initial instability, the accelerometer bias estimate appears to remain stable over the course of the trial, and thus, the first criteria is satisfied. However, the bias estimate fails to remain within the two-sigma bounds of the variance estimate, indicating that the process noise is not adequately tuned for these states. As a result, the filter does not properly

adjust the estimate as new measurements are encountered. The z-axis bias estimate appears to recover from the initial offset by the end of the trial, but the other axes drift outside the two-sigma bounds. It is possible that this offset is related to the decreased velocity estimate variance in Figure 4.10 since it occurs in the same axis. Further work is required to investigate the root cause of the issue.

Due to the initial offset and drift, the accelerometer bias estimate fails the second qualitative criteria. While this failure indicates incorrect operation, it is still possible for other filter states to achieve these criteria. The results already presented in this section demonstrate that the filter appears functional despite partial failure of these bias states. The relatively low noise and bias instability of the selected IMU likely mitigate any incorrect tuning or modeling to achieve these results. The last qualitative criterion is not applicable for any of the inertial sensor bias states because no direct measurements of these states are available. No measurement innovation can be computed in either case.

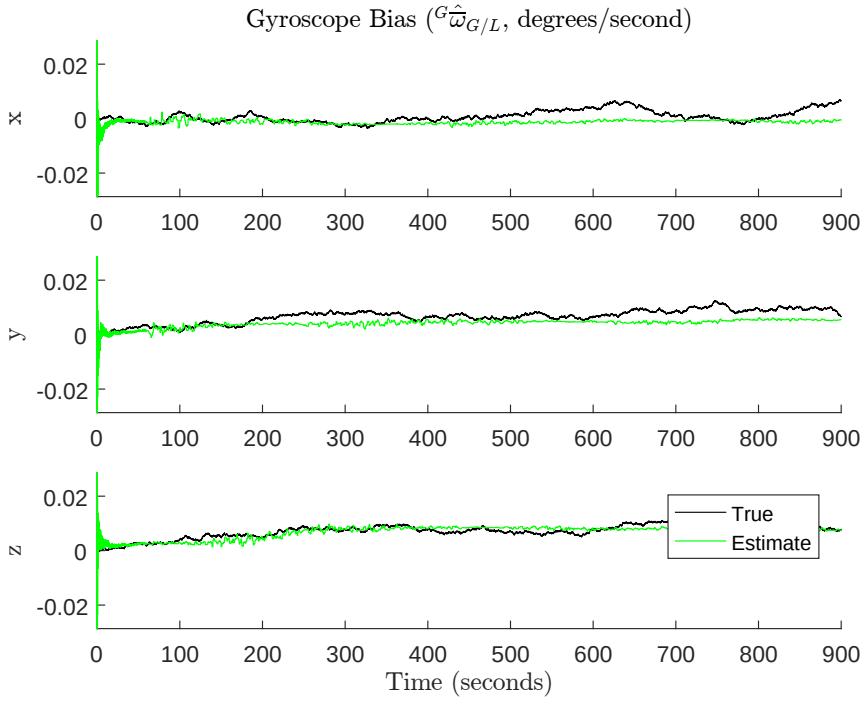

Figure 4.18: Random Trajectory - Gyroscope Bias

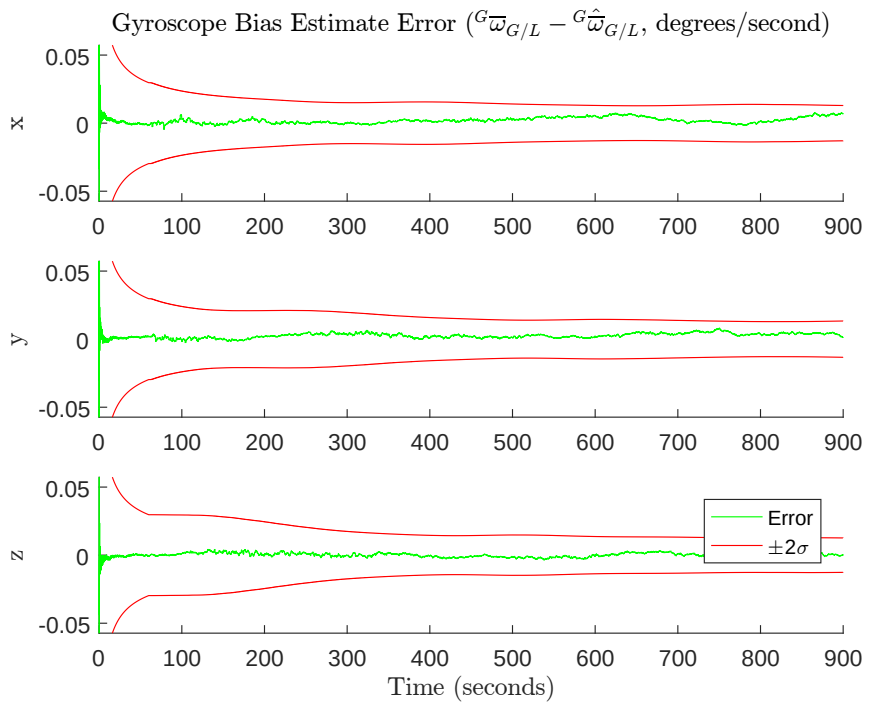

Figure 4.19: Random Trajectory - Gyroscope Bias Estimate Error

The gyroscope bias state and estimate error are shown in Figure 4.18 and Figure 4.19. The gyroscope bias estimate experiences the same initial instability since a correct startup value is unavailable. However, this bias does not exhibit the same initial offset. As a result, the estimate error appears to have a mean of approximately zero with notable stability. The estimate also clearly remains within the two-sigma bounds of the estimate error. The gyroscope bias estimate, thus, satisfies the first two criteria. This comparatively mild behavior is partially due to the independence of the gyroscope bias state. In the process model, coupling of this bias state is minimal: the gyroscope bias influences attitude but does not directly influence the acceleration, velocity, or position states. Therefore, implicit coupling of the bias within the filter is minimized. Coupling of the gyroscope bias increases when the installation offset of the accelerometer and rotation of the earth are included in the kinematics model. As with the accelerometer bias, the third criterion is not applicable in this case either since the gyroscope bias is not directly measured.

4.3.2 Trial Comparison

Comparison of data among random trajectories is performed by computing the total root-mean squared (RMS) and final state errors for each trial. The RMS state errors, $\epsilon_{RMS,i}$, are computed as shown in Equation 4.2, where i denotes the trial index, k denotes the time index, and M denotes the number of samples in the trial.

$$\boldsymbol{\epsilon}_{RMS,i} = \sqrt{\frac{1}{M}\sum_{k=0}^{M-1}(\hat{\boldsymbol{x}}_{i,k} - \boldsymbol{x}_{i,k})^2} \tag{4.2}$$

The primary purpose for this comparison is to ensure insensitivity to specific realizations of noise. Such sensitivity is manifest through significant inconsistencies in the RMS and final errors among trials. These values are compared through inspection of histograms for each state.

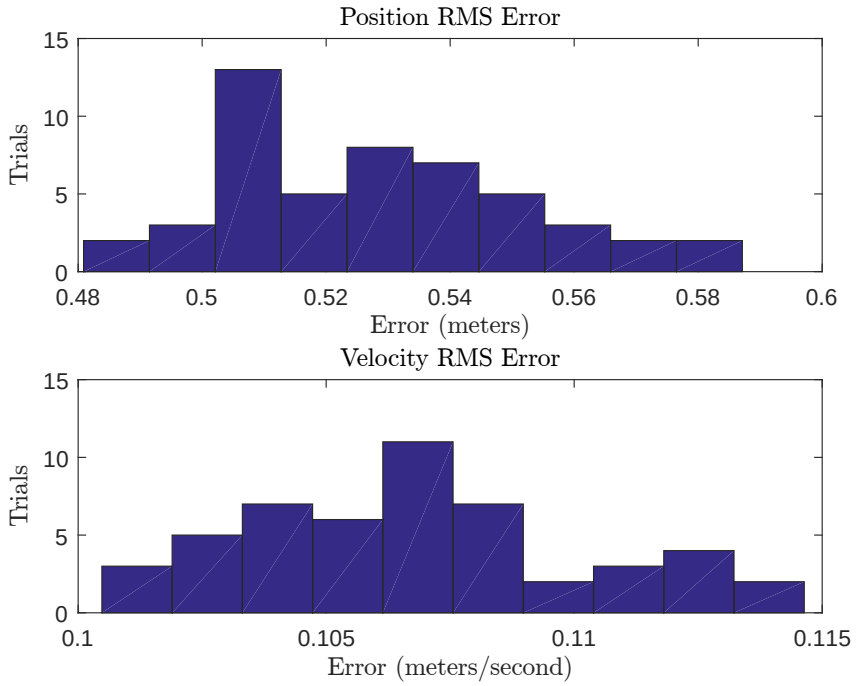

Figure 4.20: Random Trajectory - Total RMS Error - Position/Velocity

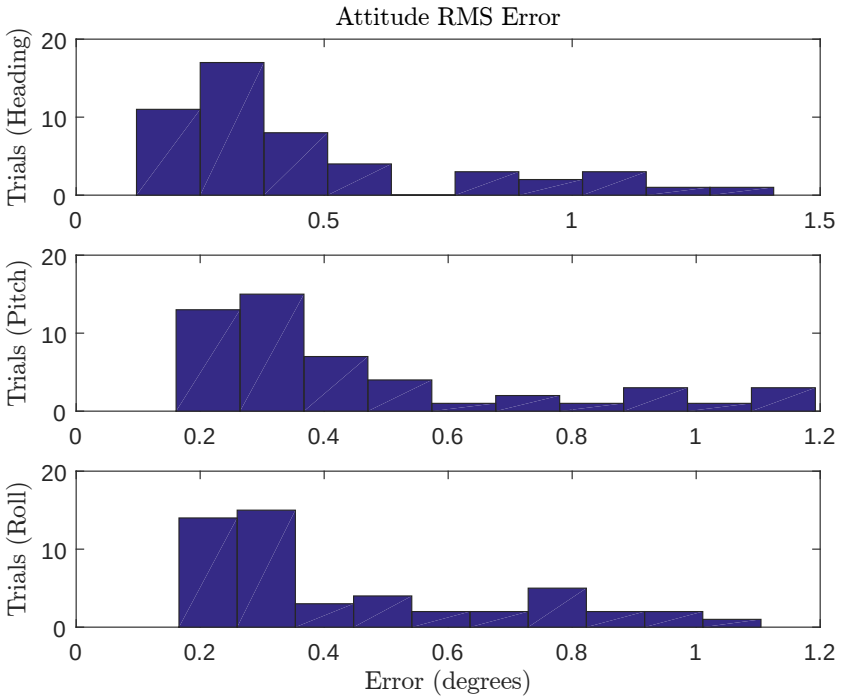

Figure 4.21: Random Trajectory - Total RMS Error - Attitude (Tait-Bryan Angles)

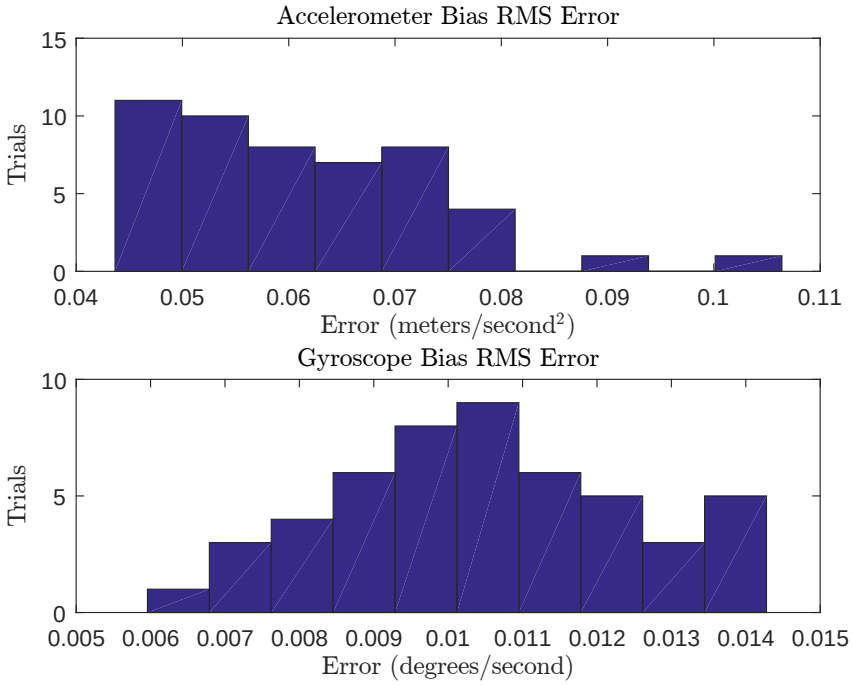

Figure 4.22: Random Trajectory - Total RMS Error - Biases

Component errors for vector states like position, velocity, and the inertial sensor biases are reduced to radial errors for simplicity of comparison. The quaternion errors are converted to Tait-Bryan angles and compared in component form to preserve their independence since uncertainties in heading may differ significantly from those of pitch and roll [10]. In a truly GNSS-aided system, reduction of the position and velocity states to their respective norms would also be discouraged since GNSS demonstrates different horizontal and vertical accuracies. Since these accuracies are not effectively modeled for this research, reduction through the radial error computation is possible. The results of this comparison are given by the histograms in Figure 4.20, Figure 4.21, and Figure 4.22. These figures demonstrate that no significant outliers are present in the test of fifty trials since the distributions are relatively consistent. Adjusting bin widths does not appreciably change the distribution.

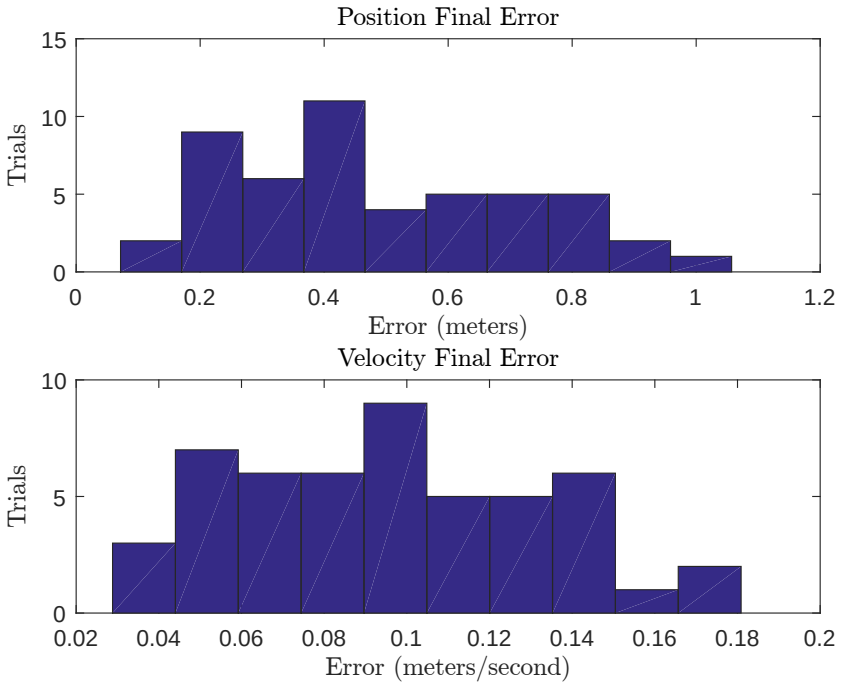

Figure 4.23: Random Trajectory - Final Error - Position/Velocity

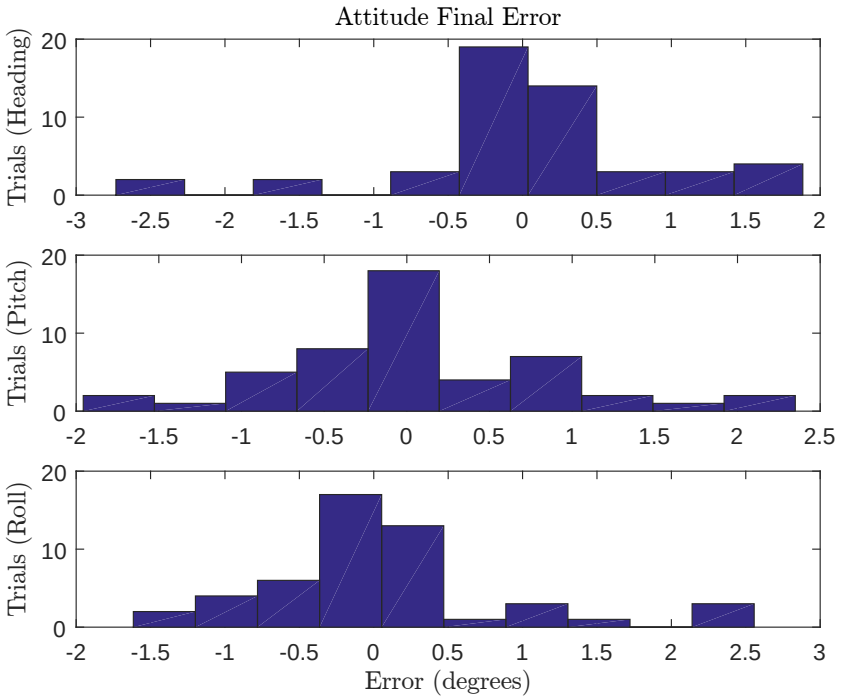

Figure 4.24: Random Trajectory - Final Error - Attitude (Tait-Bryan Angles)

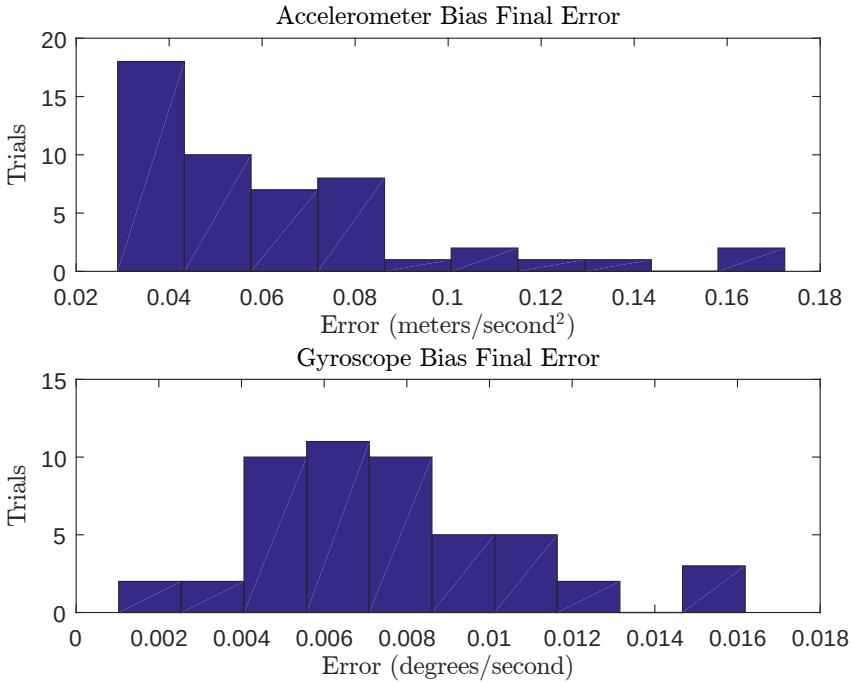

Figure 4.25: Random Trajectory - Final Error - Biases

To provide another measure of consistency, the final state errors are tabulated in a similar form. The resulting histograms are given in Figure 4.23, Figure 4.24, and Figure 4.25. These figures demonstrate the same result: the filter delivers relatively consistent performance for each trial. Like the previous distributions, the final error distributions do not change significantly for smaller bin widths. Given the overall results from random trajectory testing, the filter appears to demonstrate adequate stability and consistency of operation to begin testing with more representative trajectory data.

4.4 X-Plane Simulation

The second series of tests performed on the filter utilizes X-Plane to provide more realistic vehicle trajectory data. All simulated flights were performed in the Beechcraft King Air C90. While X-Plane natively supports flight data logging, the log rate is far too low to achieve the required sample interval, $T_{s,sim}$. Data are acquired from the simulator at a higher rate using a custom plugin written in C++. This plugin performs basic buffering and storage of flight data at the maximum available rate, the rate of the X-Plane flight model itself. This method of data retrieval is typically discouraged due to its impact on the overall simulation speed. However, the plugin limits high-rate operations to efficient data buffering, periodically flushing the buffer to disk. In testing, the plugin demonstrates negligible performance loss and does not adversely affect piloting. Despite the logging rate increase facilitated by the plugin, the maximum rate is still below the desired simulation rate and data are output at irregular sampling intervals. Interpolation is required to upsample the data to the desired simulation rate and regularize the sampling interval. The interpolation includes simple discontinuity checks to detect abnormalities in the post-processed trajectory. The filter is tested in a Monte Carlo simulation using the interpolated X-Plane trajectory to generate sensor data. A total of thirty trials are performed, each with different realizations of sensor noise and bias walk. This section first presents results of a single trial and then discusses the combined results from multiple trials.

4.4.1 Single Trial

Results for a single X-Plane trial closely resemble those from random trajectory testing. The filter is again tested against the three qualitative criteria of filter stability, accurate covariance estimation, and uncorrelated measurement residuals. It is important to note that figures involving X-Plane time series data do not start at zero because the X-Plane time epoch occurs during program startup and does not reset with logging.

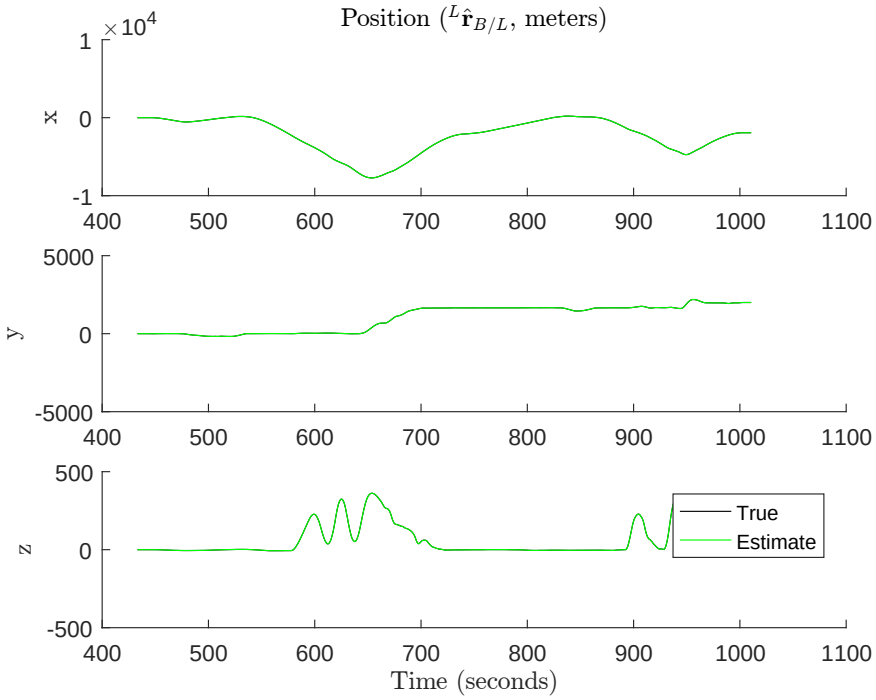

Figure 4.26: King Air C90 Simulation - Position
(True and estimated values are coincident at this scale)

The position state and estimate errors are shown in Figure 4.26 and Figure 4.27. Similar to the random trajectory data, the axis extents required to view the full trajectory cause the true and estimated trajectories to appear almost perfectly coincident. The estimate error again provides a more useful perspective. Upon inspection, the position estimate appears stable as before, but struggles to remain inside the two-sigma bounds for 95 percent of the

trial. This difficulty is due to the more aggressive trajectory and oscillatory pilot inputs injected by the X-Plane simulation.

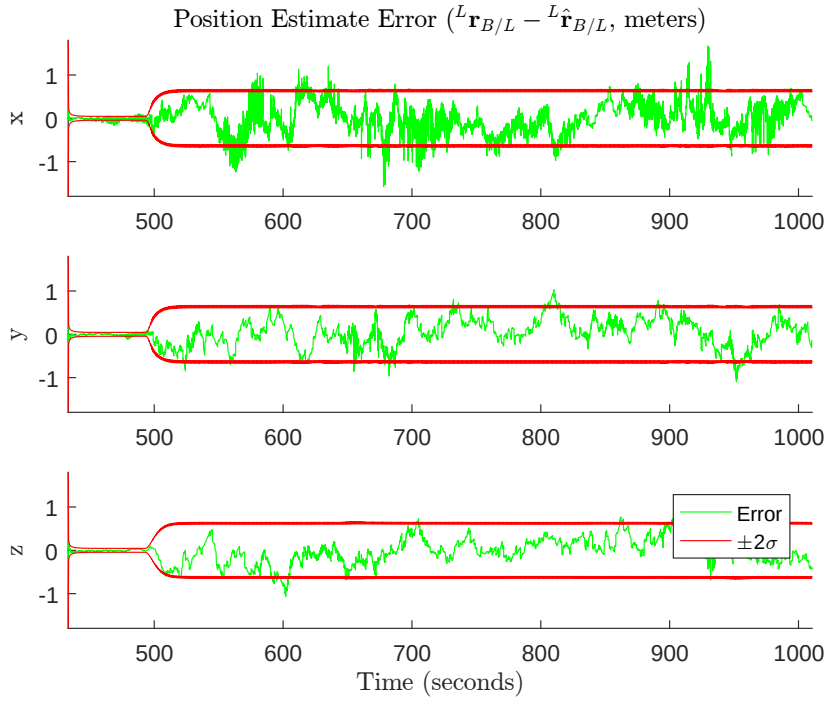

Figure 4.27: King Air C90 Simulation - Position Estimate Error

One important observation from Figure 4.27 and enlarged in Figure 4.28, is that the x-axis estimate error contains a large noise component compared to that of the random trajectory simulation shown in Figure 4.6. Although less noticeable, the other two axes contain short intervals of noisy behavior as well. In the ideal case, the position estimate should consist of piecewise quadratic segments due to double integration of the accelerometer measurements. The expected behavior is observed for the random trajectory simulations, as in Figure 4.6. Since this noisy behavior is not observed in the random trajectory simulation, it must be the result of some difference between the two simulation types. Upon initial inspection, the X-Plane acceleration and angular rate data appear to contain noisy regions, like those shown in Figure 4.2 and Figure 4.3.

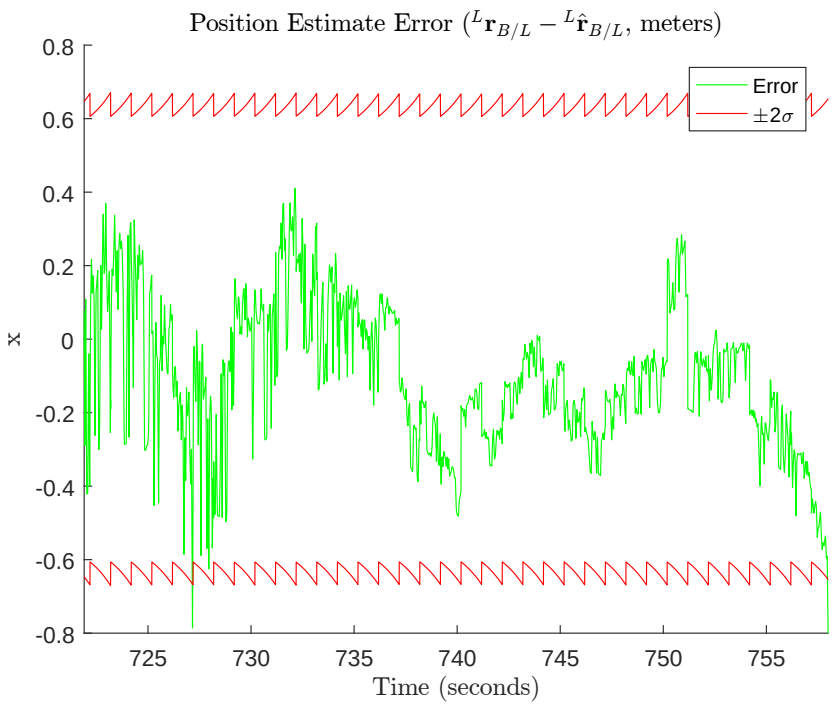

Figure 4.28: King Air C90 Simulation - Position Estimate Error - Enlarged

Viewed at an appropriate scale, these regions are actually sinusoidal in nature, with an approximate frequency of 1 Hz. Although the exact cause of the increased noise in the x-axis position estimate error is unknown, it is possible that this noise is the result of oscillating pilot input, potentially interacting with some sort of instability within the system model. Further troubleshooting work is required to determine the root cause of this noise.

The position measurement residual and auto-correlation in Figure 4.29 and Figure 4.30 demonstrate similar performance to the random trajectory test case, yielding low correlation in the residual. The position estimate appears to satisfy the stability and uncorrelated measurement residual criteria, but fails to properly estimate the position estimate error covariance. It is possible further tuning would alleviate this additional drift.

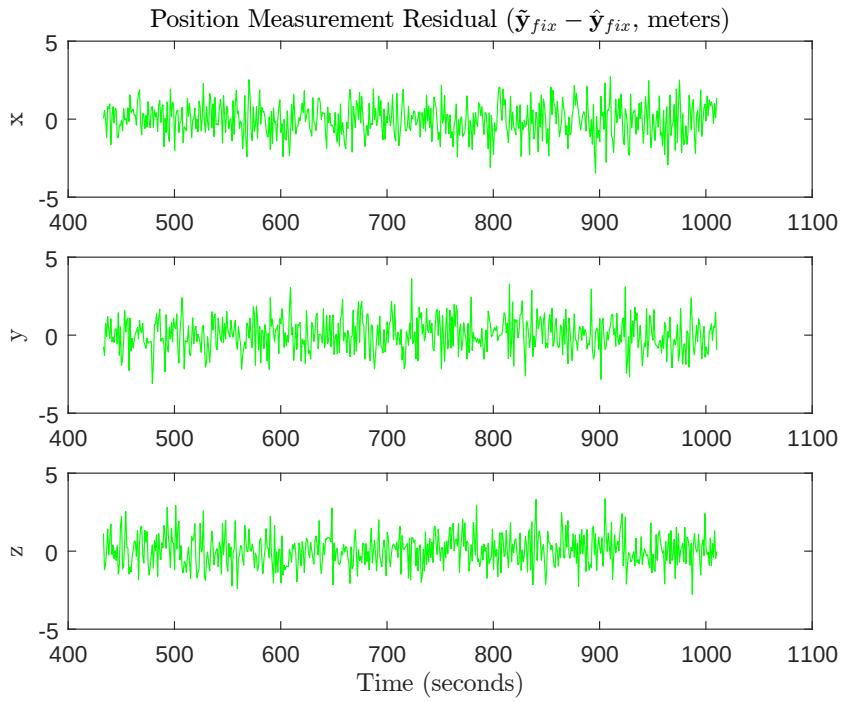

Figure 4.29: King Air C90 Simulation - Position Measurement Residual

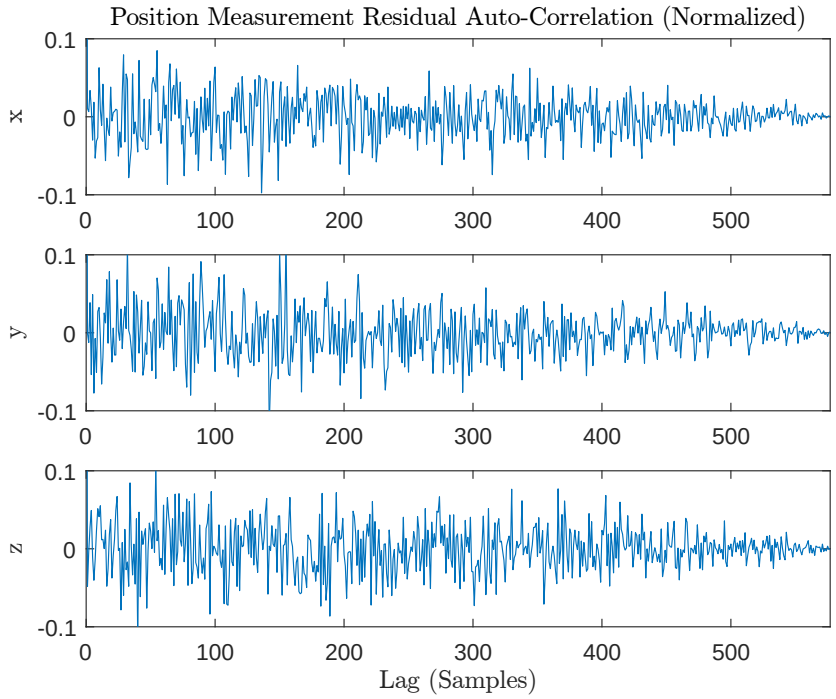

Figure 4.30: King Air C90 Simulation - Position Measurement Residual Auto-Correlation

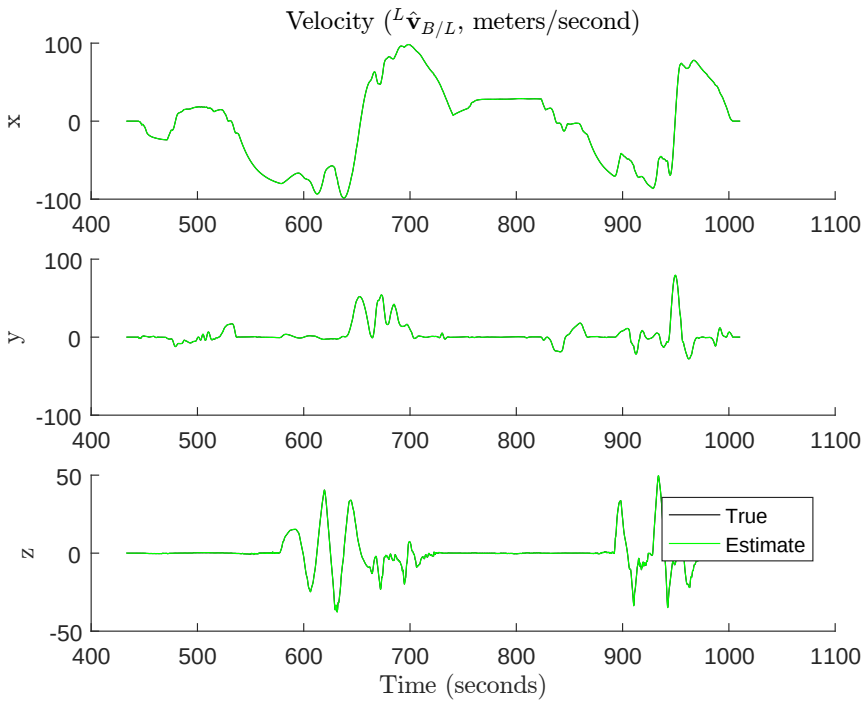

Figure 4.31: King Air C90 Simulation - Velocity
(True and estimated values are coincident at this scale)

The velocity estimate, shown in Figure 4.31 and Figure 4.32, exhibits much better performance. The estimate stability appears adequate, and the estimate error remains within the two-sigma bounds for most of the trial. The ripple present in the velocity estimate error variance indicates that prediction and measurement updates are driving changes to the covariance rather than the manually-tuned process noise covariance. This mixture of contributions is unlike the corresponding estimate for position, which appears primarily driven by the input process noise covariance. These results suggest that the velocity estimate meets the first two qualitative criteria. In Figure 4.33 and Figure 4.34, performance of the velocity measurement residual is also quite similar to the random trajectory test. Given the low auto-correlation, the velocity estimate satisfies the uncorrelated measurement residual criterion.

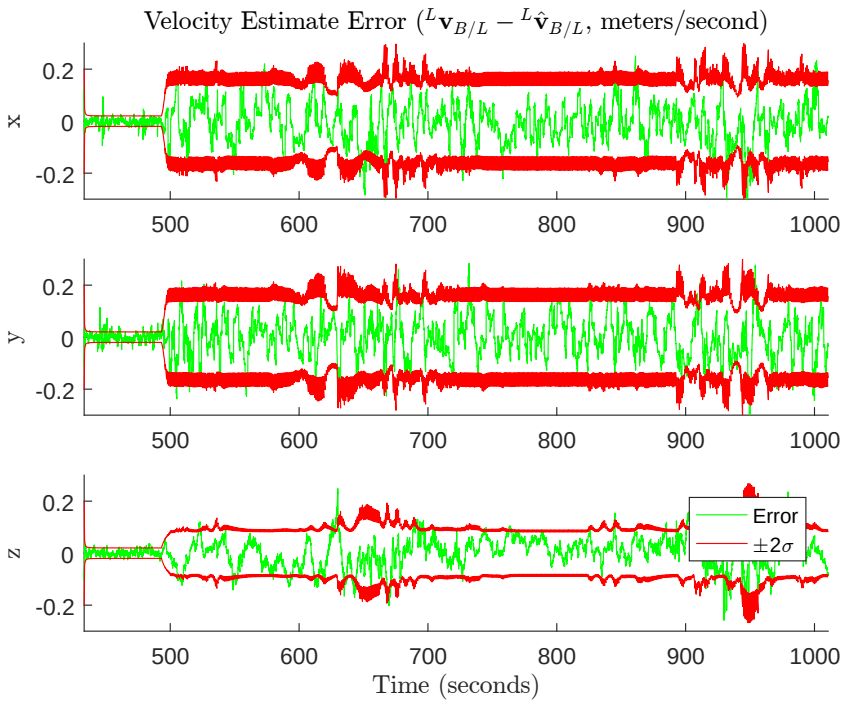

Figure 4.32: King Air C90 Simulation - Velocity Estimate Error

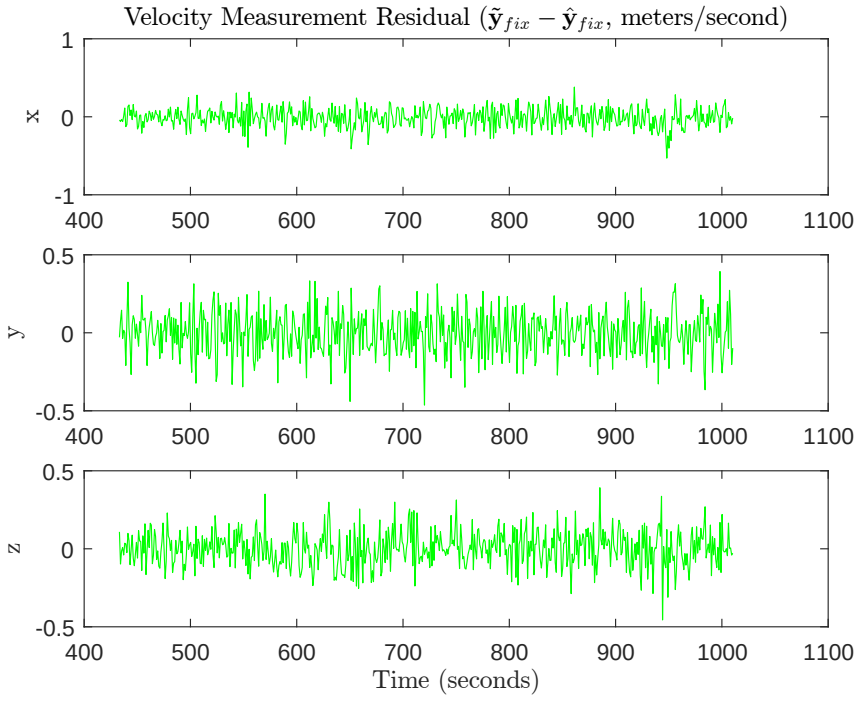

Figure 4.33: King Air C90 Simulation - Velocity Measurement Residual

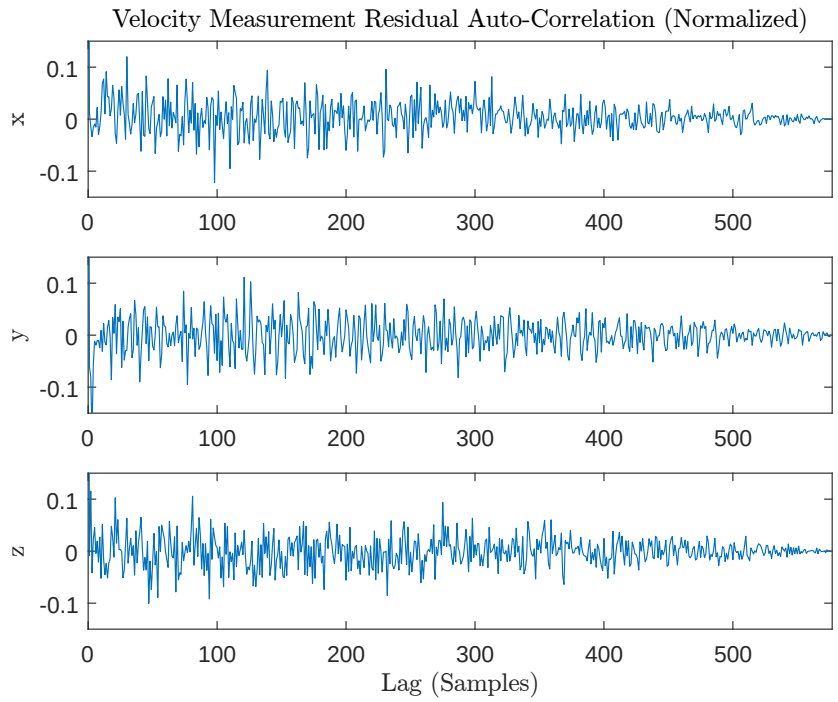

Figure 4.34: King Air C90 Simulation - Velocity Measurement Residual Auto-Correlation

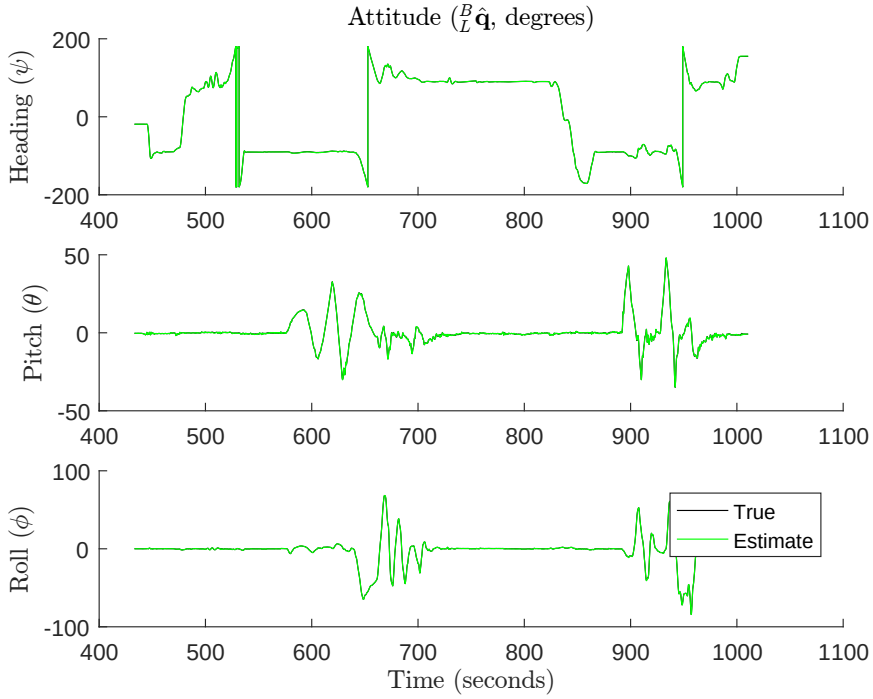

Figure 4.35: King Air C90 Simulation - Attitude (Tait-Bryan Angles)
(True and estimated values are coincident at this scale)

The attitude estimate and error, in Figure 4.35 and Figure 4.36, also behave similarly to the random trajectory test results. However, the attitude estimate error variance lacks the discontinuities previously present in Figure 4.14, likely due to the absence of wrap around in the pitch and roll axes. Plots of the quaternion attitude estimate error are omitted in this section due to the lack of unusual behavior in the Tait-Bryan representation.

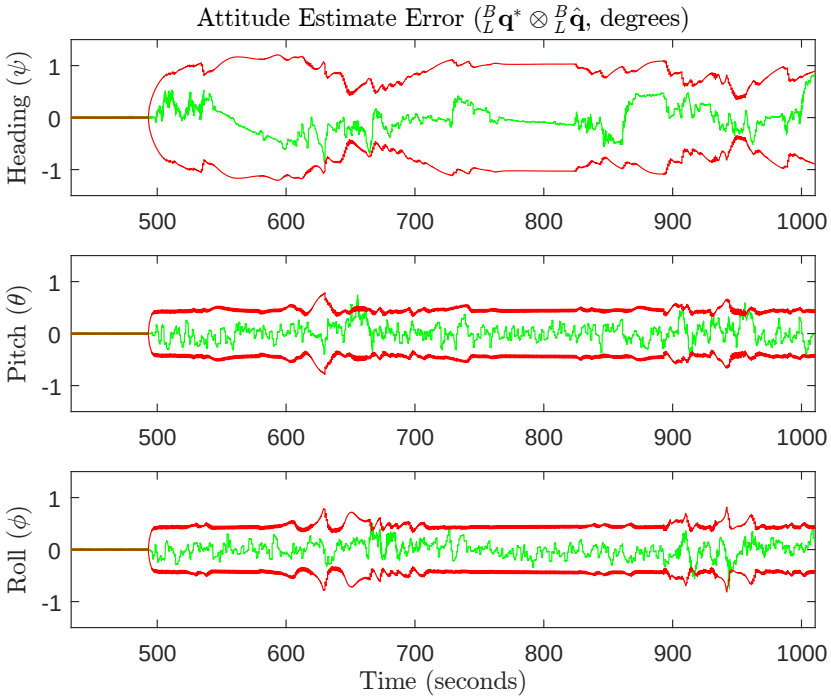

Figure 4.36: King Air C90 Simulation - Attitude Estimate Error (Tait-Bryan Angles)

The heading estimate error variance in Figure 4.36 is much larger than that of the other two axes. This difference occurs because heading is technically underdetermined in this case. Pitch and roll are implicitly related to accelerometer measurements by the filter through accounting of the gravity vector. In this model, gravity is defined as perpendicular to the local level, and thus, changes in heading do not affect measurement of the gravity vector. Therefore, the implicit relationship between heading and the inertial measurements is weaker than it is for the pitch and roll axes, explaining the higher estimated error in this

axis. The estimated error is also smoother in heading since it has a less direct relationship to the noisy inertial sensor measurements. Given its overall behavior, the attitude estimate again satisfies the two applicable qualitative criteria.

Like the other states so far, the accelerometer bias behavior matches that of the random trajectory testing. The bias states and estimates are provided in Figure 4.37 and Figure 4.38. These states demonstrate similar initial instability and a similar estimate offset in the z axis. The fact that the offset occurs during both trajectory simulations indicates the underlying cause must be a modeling or tuning error. Interestingly, in Figure 4.37, the estimated bias waveform closely matches that of the true bias state despite the offset. Due to this offset, the accelerometer bias estimate variance fails to accurately estimate the error, though the bias estimate succeeds in remaining stable over the course of the trial. As a result, the bias estimate meets the filter stability criterion, but fails the covariance estimate accuracy criterion. As before, the filter still produces reasonable estimates for the other states.

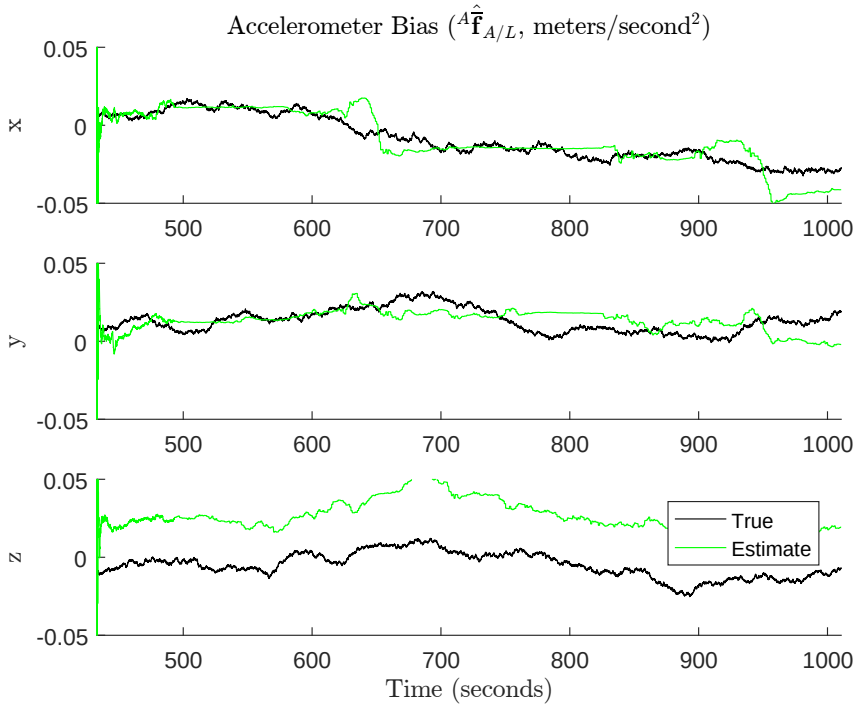

Figure 4.37: King Air C90 Simulation - Accelerometer Bias

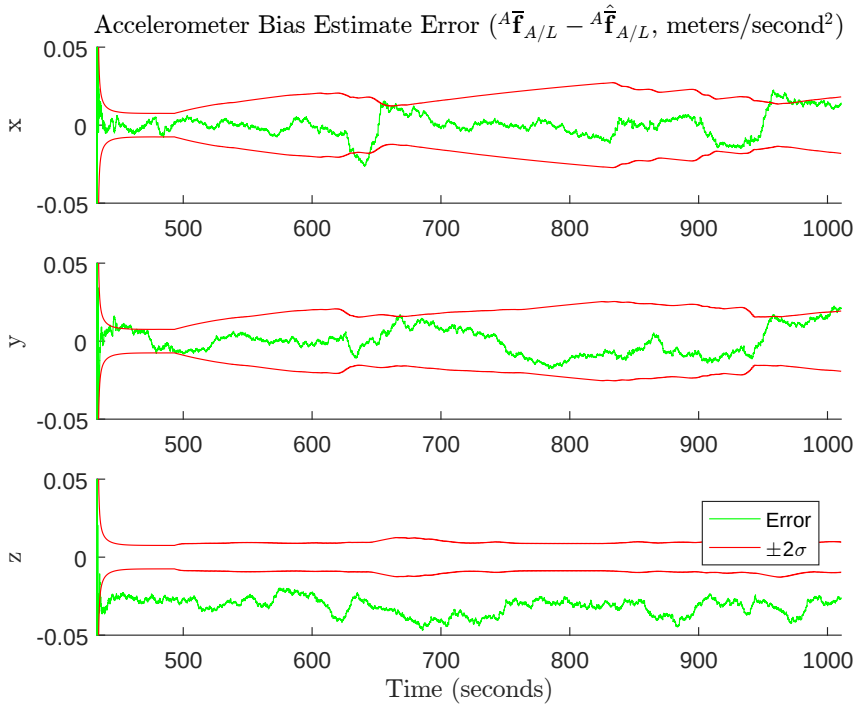

Figure 4.38: King Air C90 Simulation - Accelerometer Bias Estimate Error

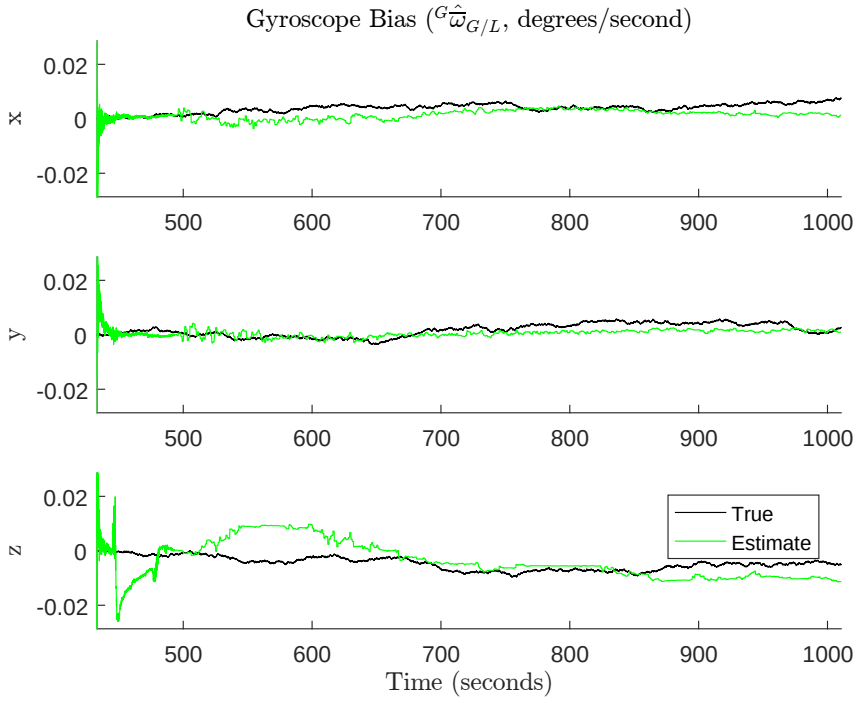

Figure 4.39: King Air C90 Simulation - Gyroscope Bias

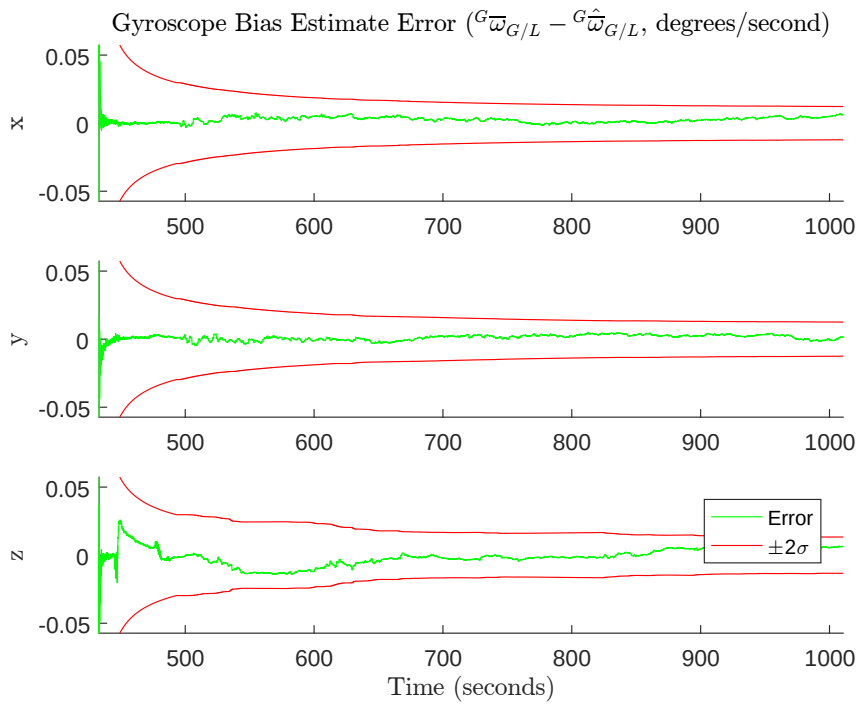

Figure 4.40: King Air C90 Simulation - Gyroscope Bias Estimate Error

The gyroscope bias estimate and corresponding estimate error, shown in Figure 4.39 and Figure 4.40, perform as expected considering the behavior of the other states. The gyroscope bias estimate experiences slightly more initial instability with about the same rate of convergence for the estimate variance. Despite this initial difficulty, the bias estimate remains stable and within the two-sigma estimate error bounds, clearly satisfying the two applicable criteria. Given the overall comparable performance of this single trial to the random trajectory simulation, the analysis continues on to evaluate the Monte Carlo results.

4.4.2 Trial Comparison

Comparison of the X-Plane trajectory trials follows the same basic approach as that of the random trajectory trials. The comparison begins with a cursory analysis of the total RMS state estimate error distributions to establish consistency among trials. The final state estimate error distributions are included in Figure A.14, Figure A.15, and Figure A.16 for

completeness, but omitted from this section for brevity since the results are similar. Since the same X-Plane trajectory is used for all Monte Carlo trials, the state estimate errors are also combined using RMS and analyzed as an ensemble result.

The total state estimate RMS error distributions are shown in Figure 4.41, Figure 4.42, and Figure 4.43. As before, position, velocity, and bias RMS errors are combined while the attitude RMS errors are left in component form to avoid loss of information. Based on these distributions, the filter performance again appears consistent among trials, with no significant outliers.

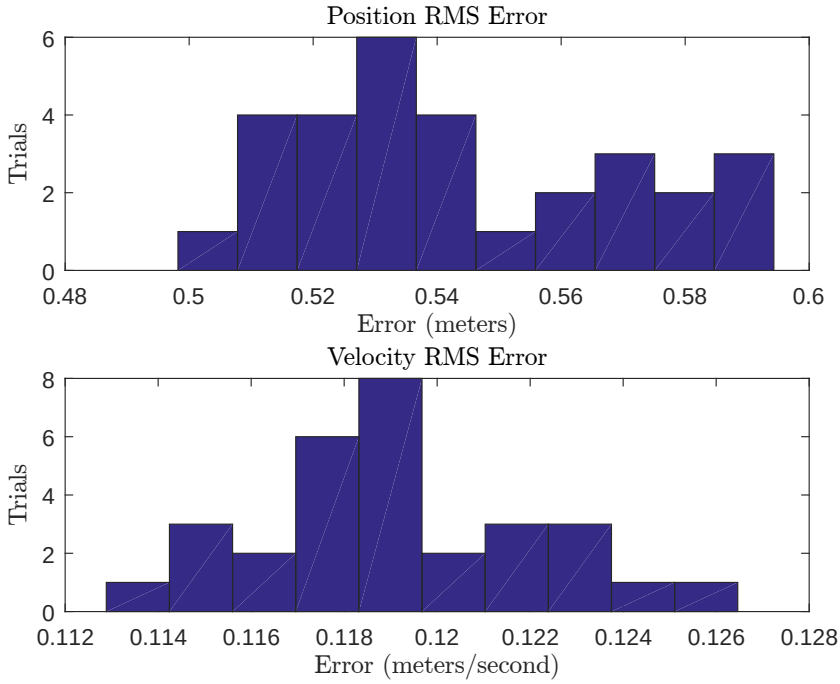

Figure 4.41: King Air C90 Simulation - Total RMS Error - Position/Velocity

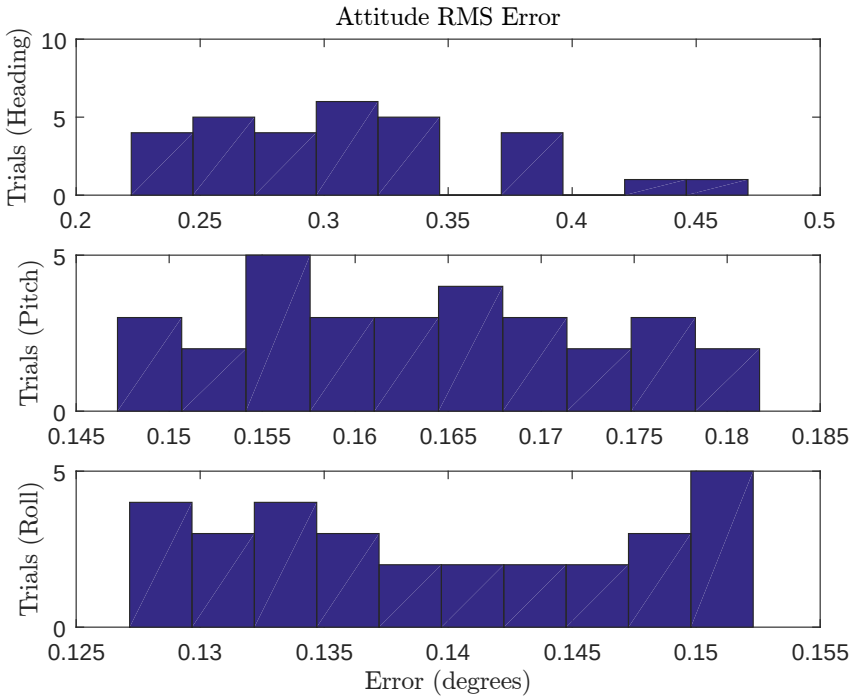

Figure 4.42: King Air C90 Simulation - Total RMS Error - Attitude (Tait-Bryan Angles)

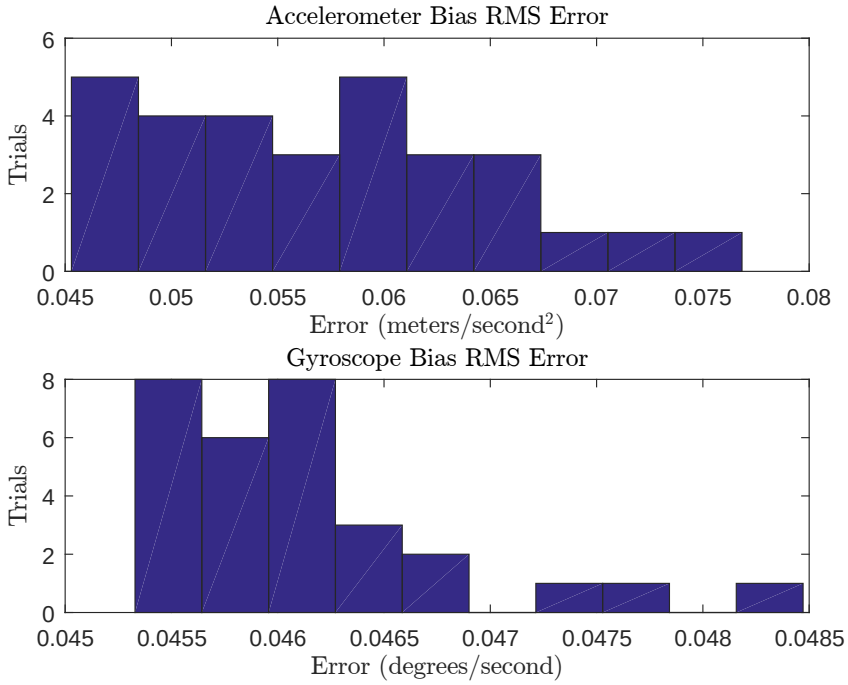

Figure 4.43: King Air C90 Simulation - Total RMS Error - Biases

To continue with the analysis, time series data for the state estimate error are combined among all trials in the RMS sense to generate a single time series plot of state estimate RMS error for the Monte Carlo simulation. These results provide a combined sense of the system accuracy over many realizations of measurement noise and sensor biases. Combination of the time series data is performed according to Equation 4.3, where L denotes the number of trials, i denotes the trial index, and k denotes the time index.

$$\boldsymbol{\epsilon}_{RMS,k} = \sqrt{\frac{1}{L}\sum_{i=0}^{L-1}(\hat{\boldsymbol{x}}_{i,k} - \boldsymbol{x}_{i,k})^2} \tag{4.3}$$

The combined position estimate error is given in Figure 4.44. This figure provides a different perspective of the initialization process. Filter initialization is marked by a rapid initial decrease in estimate error, followed by a region of minimum estimate error.

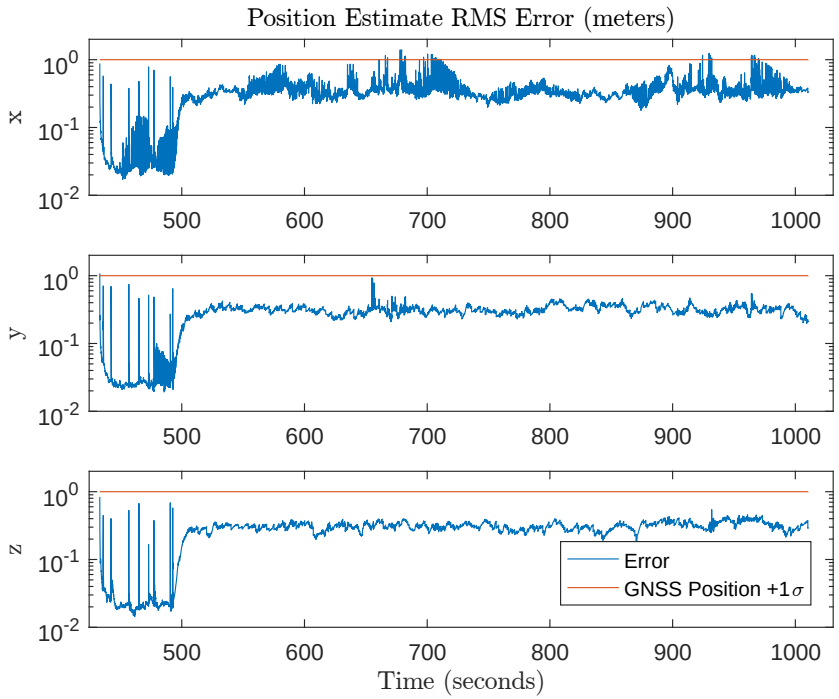

Figure 4.44: King Air C90 Simulation - Position Estimate RMS Error

This region contains several large transients, presumably due to the initial instability caused by the incorrect startup values of the bias states. At the end of initialization, the error increases from this minimum to the value afforded by the GNSS measurement updates. The x-axis and initial y-axis data exhibit the same noise seen previously in Section 4.4.1. The presence of this noise in the combined plot indicates that the noise exists in a large number of the Monte Carlo trials and not the result of sensitivity to a specific noise realization. Overall, the position estimate error remains below one standard deviation of the simulated GNSS measurements, depicted in Figure 4.44. This result demonstrates that the filter produces a superior estimate to using GNSS measurements alone.

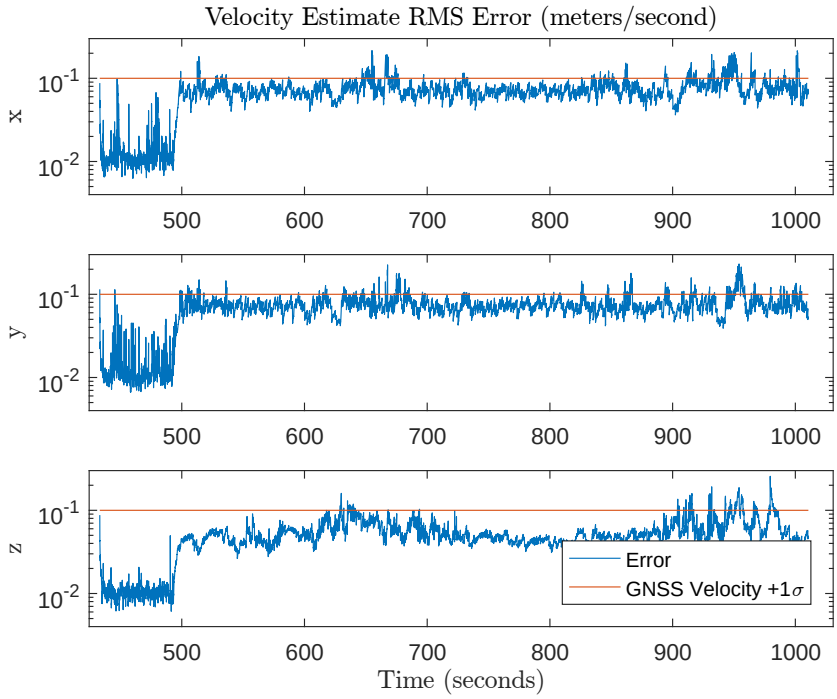

Figure 4.45: King Air C90 Simulation - Velocity Estimate RMS Error

The velocity estimate in Figure 4.45 exhibits the same behavior with respect to initialization and the start of normal filter operation. Transients during initialization are more significant in the velocity states, although the z axis appears relatively unaffected. This

difference during initialization may account for the improved performance of the z-axis velocity covariance estimate. As a result, the z-axis error is much lower than the other two axes, but also experiences more overall variability. Further investigation is required to determine the precise cause of these differences. Like position, the velocity estimate remains mostly below one standard deviation of the simulated GNSS velocity measurements. This result is consistent with the expectation of correct filter operation.

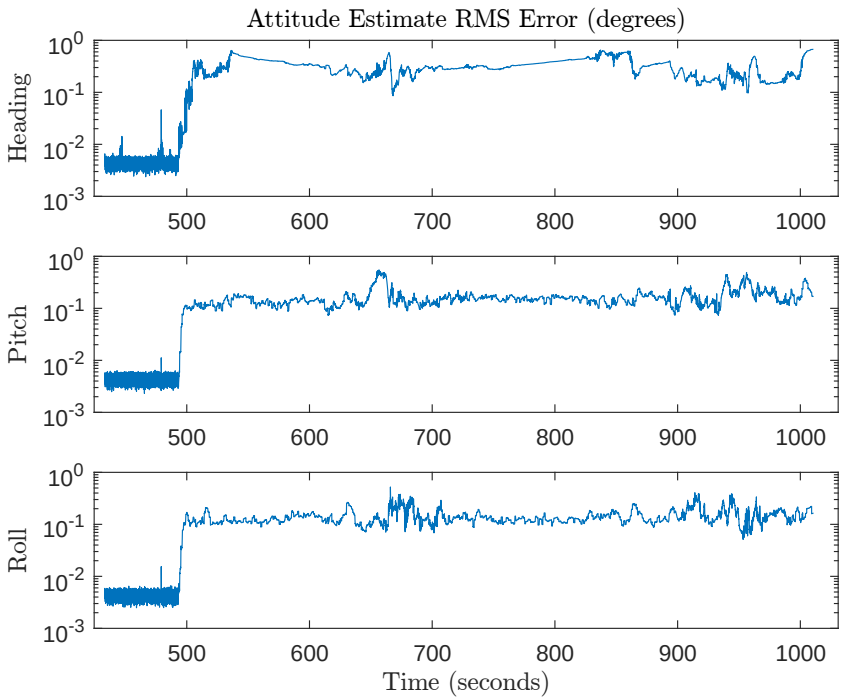

Figure 4.46: King Air C90 Simulation - Attitude Estimate RMS Error (Tait-Bryan Angles)

The combined attitude estimate error in Figure 4.46 does not appear to suffer transients in initialization to the same degree as position and velocity. The few transients that exist during this period appear heavily correlated with pilot control inputs (see Section A.2), as do many of the other regions of variability. Unlike the position and velocity errors, the attitude estimate error exhibits more regions of correlated behavior because no direct

measurement of attitude is available outside of initialization. Nevertheless, the attitude estimate demonstrates reasonable performance during the region of normal filter operation.

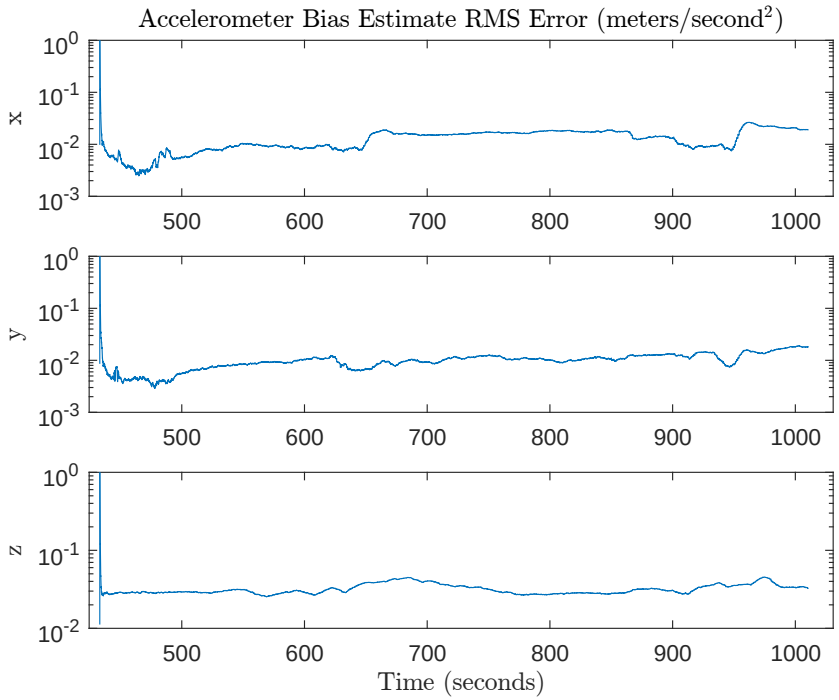

Figure 4.47: King Air C90 Simulation - Accelerometer Bias Estimate RMS Error

The accelerometer and gyroscope bias states in Figure 4.47 and Figure 4.48 are much less active and do not vary significantly between initialization and normal filter operation. The startup instabilities are present in all axes as each experience a large overshoot, followed by the expected convergence of the estimate. The z axis of the accelerometer bias converges much more quickly than the other two axes. This behavior is not shared by the z axis of the gyroscope bias, which experiences a large offset that eventually converges back to the baseline value of the error. Interestingly, the anomalous behavior occurs in the z axis of both biases. However, based on the time of occurrence, the likelihood of a common root cause is reduced.

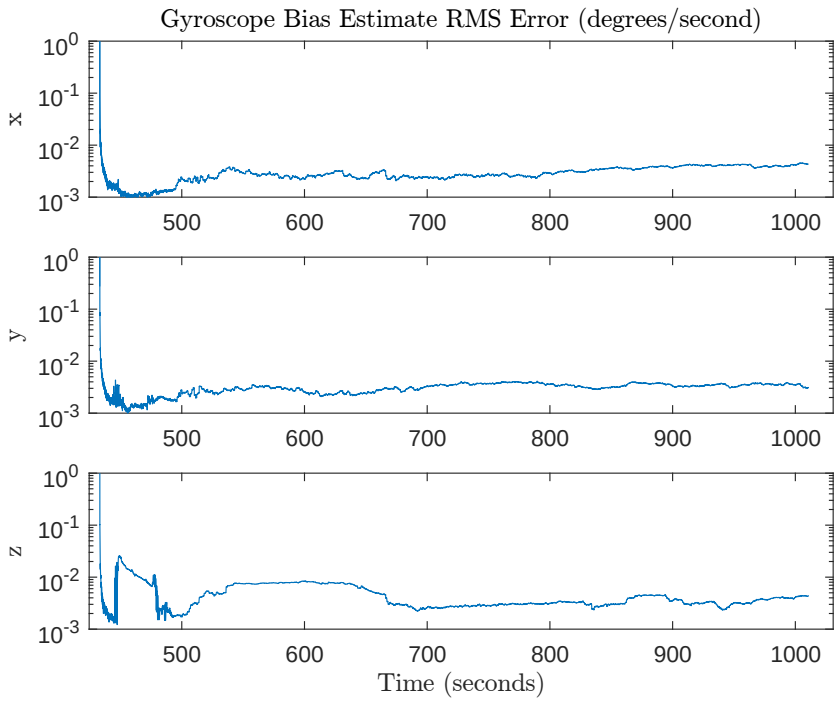

Figure 4.48: King Air C90 Simulation - Gyroscope Bias Estimate RMS Error

Given the results presented in this section for a variety of test cases and noise realizations, the modularized filter appears to be operating consistently despite relatively poor tuning. All of the estimated filter states remain stable throughout these test cases, and the position and velocity measurement residuals are reasonably uncorrelated. The filter, however, fails to achieve accurate covariance estimation in the accelerometer bias states. As a result, the filter does not completely satisfy all three qualitative criteria for operation. The variety of issues encountered in the analysis merit further investigation to achieve full operation.

CHAPTER 5
CONCLUSION

This thesis presented the design of a modular navigation filter framework tailored for use in applications favoring maximal flexibility and reduced reconfiguration workload. The design was initially approached through a survey of historical and modern navigation techniques, all of which essentially reduce to the basic operations of dead reckoning and feature-based navigation. Strict discipline was exercised to harness natural modularization in the standard Kalman filter framework and to ensure separation of filter components. The resulting design achieves a great deal of generality and flexibility. This filter framework is currently employed with different process and measurement models at EOSL in a collaborative LiDAR-based mapping application.

5.1 Improvements

While the design goals for framework modularization were achieved, the performance of the implemented filter and models produced three overall anomalies in the simulated results. These anomalies are:

1. Consistent offset in z-axis accelerometer bias estimate
2. Unexplained low z-axis velocity variance estimate
3. Excessive noise in position and velocity state estimates during X-Plane simulation

The first two issues could be the result of inadequate tuning or modeling errors. The unaugmented form of the unscented Kalman filter is incompatible with the nonlinearly coupled measurement noise generated in the inertial sensor models. It is possible the fictitious additive process noise was insufficient or unable to handle the magnitude or correlation of

the true measurement noise. The unusual noise present in the X-Plane simulation may be the result of a difference in the underlying kinematics models applied in X-Plane.

One factor that likely contributes to these issues is the method of initialization. As previously mentioned, this method generates instability in the bias estimates due to a large discrepancy between the estimated and true initial values. This issue is easily mitigated by providing an initialization process that operates outside the core filter to provide a more correct initial bias estimate. This capability is commonly achieved through simple averaging of measurements given some assumed initial system condition [19]. However, the final initialization solution should preserve inputs of position, velocity, and attitude to allow for complex initialization scenarios, such as mobile transfer of alignment from another navigation system.

From a practical implementation standpoint, the benefits of modularization also tend to obfuscate implicit assumptions and dependencies between modules if not appropriately documented. These relationships are usually more apparent in a modest-sized monolithic code base. During development of the framework, multiple model configurations existed simultaneously. Development was frequently delayed through mistaken use of component modules from different system models. Much of this filter framework was modularized at the function level in MATLAB which failed to capture these relationships. Use of object-oriented design techniques and a strong type system would better facilitate expression of model dependencies and decrease the potential for implementation errors.

5.2 Future Work

After the issues of the previous section are corrected, a number of additional features and improvements are desired. For greater utility in its intended application, the simulation would benefit from higher-fidelity aiding sensor models, simulating GNSS and/or LiDAR. From a practical standpoint, this framework also neglects important features for handling delayed measurement updates, common to GNSS receivers, or performing basic fault de-

tection and measurement rejection. Important improvements to the process model are also possible. The process model implementation could be optimized by using the quaternion error state formulation to reduce the number of required states. This change could significantly improve the performance of the unscented Kalman filter. The state augmentation approach described in [6] could be applied to better represent the actual stochastic inputs and reduce the number of sources accounted through manual tuning of the process noise covariance. To further improve process noise estimation, the maximum discretization error could be dynamically added to this covariance to more carefully account for errors in numerical integration. These improvements would correct and enhance the current implementation to enable better performance and promote future design relevance.

Appendices

APPENDIX A

ADDITIONAL RESULTS

A.1 Additional Random Trajectory Data

This section contains additional and supplemental data related to the random trajectory simulation presented in Section 4.3. The first set of figures depict the true state generated from the random walk trajectory model. The second set of figures show an enlarged view of the initialization process for the position, velocity, and attitude states.

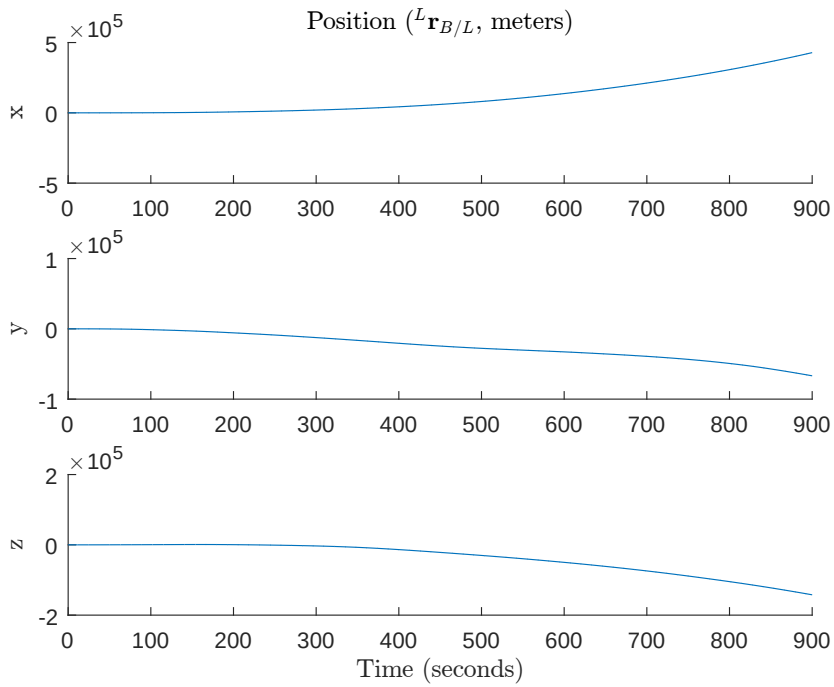

Figure A.1: Random Trajectory - Position

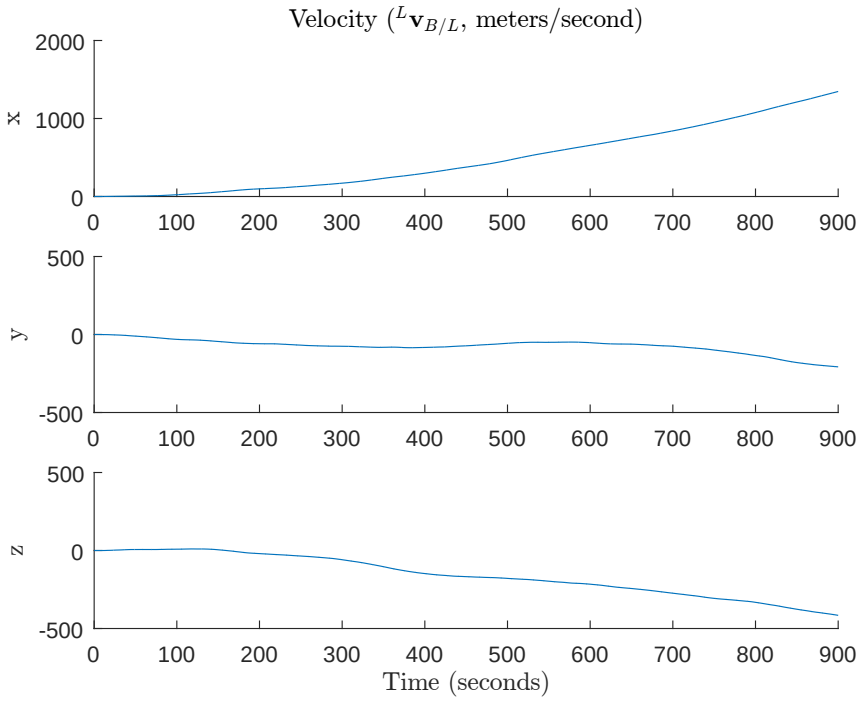

Figure A.2: Random Trajectory - Velocity

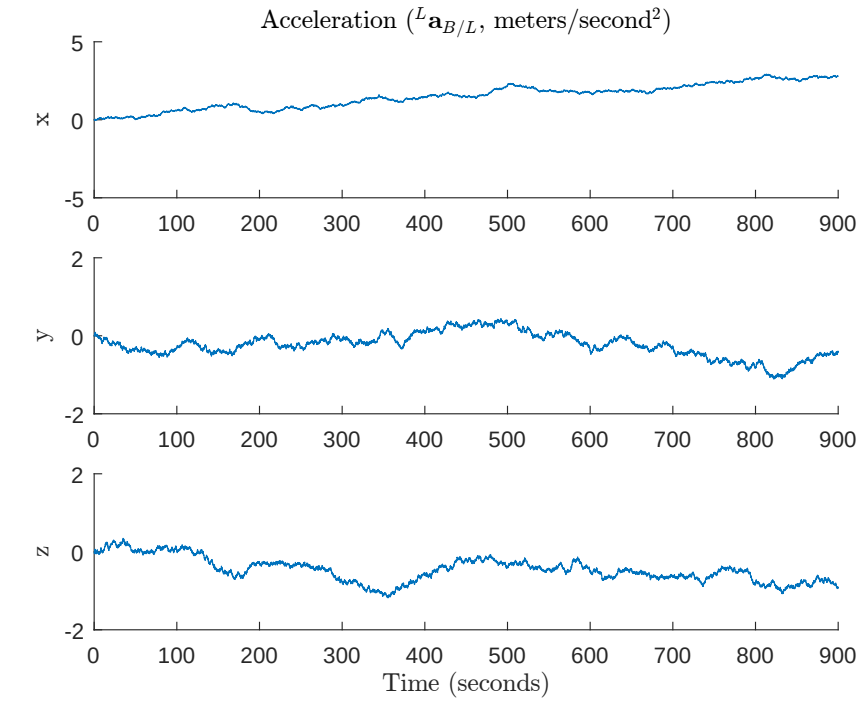

Figure A.3: Random Trajectory - Acceleration

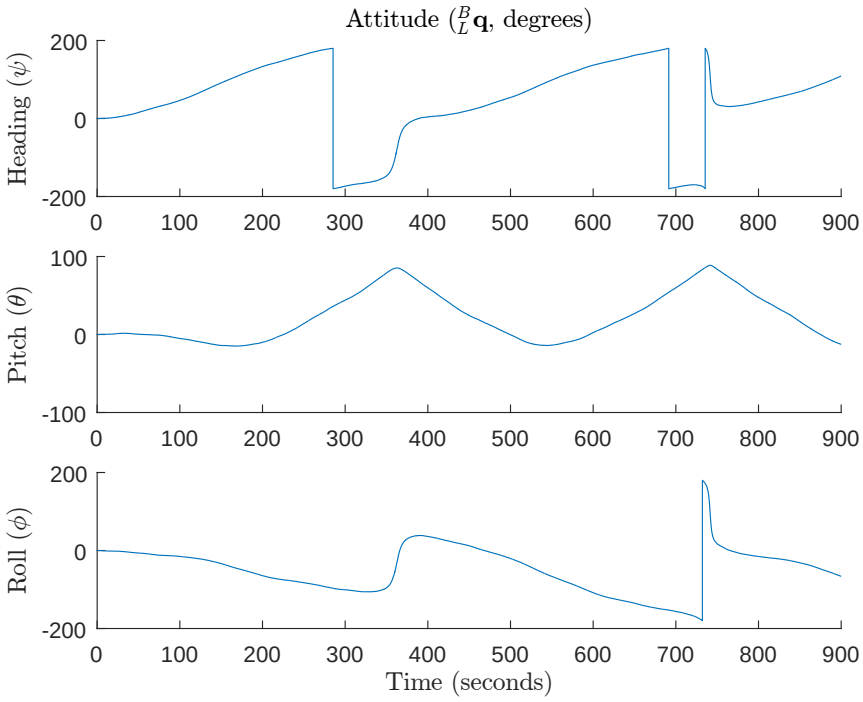

Figure A.4: Random Trajectory - Attitude (Tait-Bryan Angles)

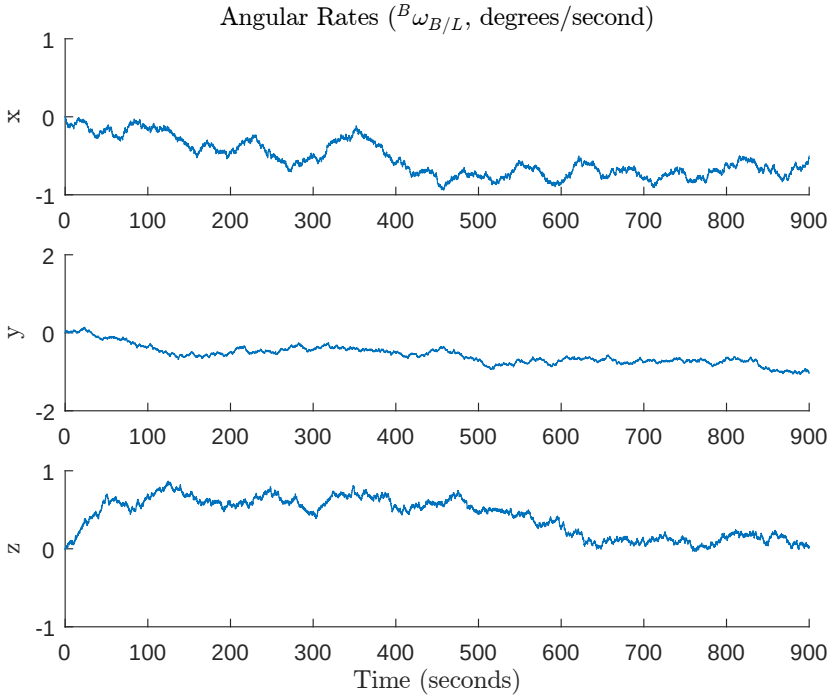

Figure A.5: Random Trajectory - Angular Rates

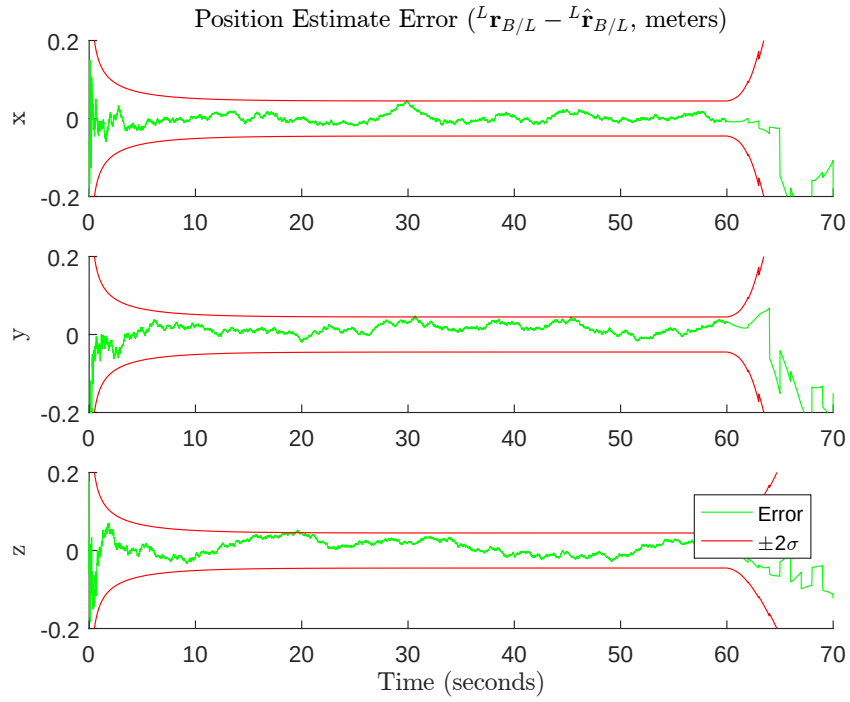

Figure A.6: Random Trajectory - Position Estimate Error - Initialization

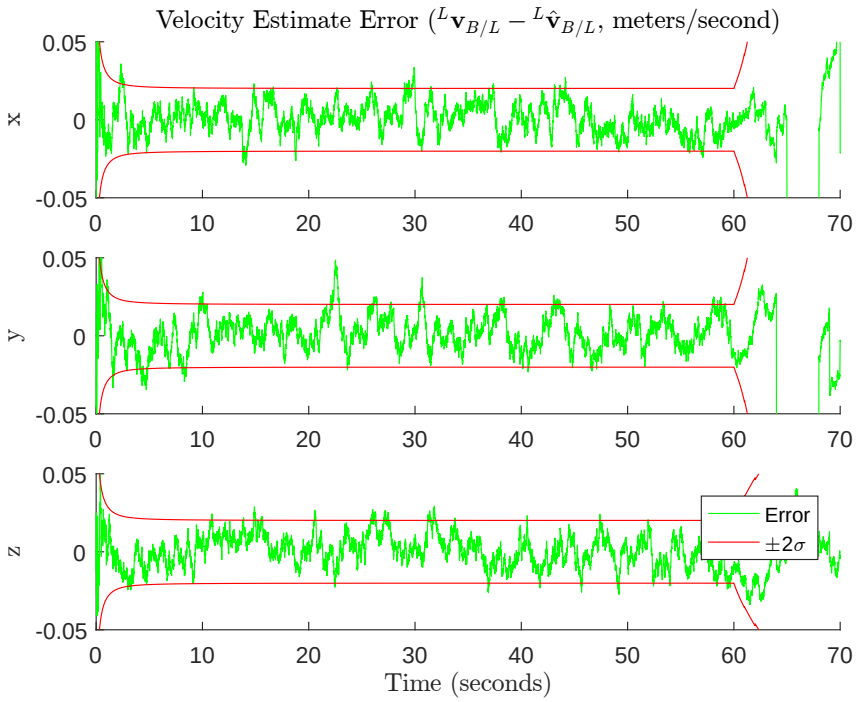

Figure A.7: Random Trajectory - Velocity Estimate Error - Initialization

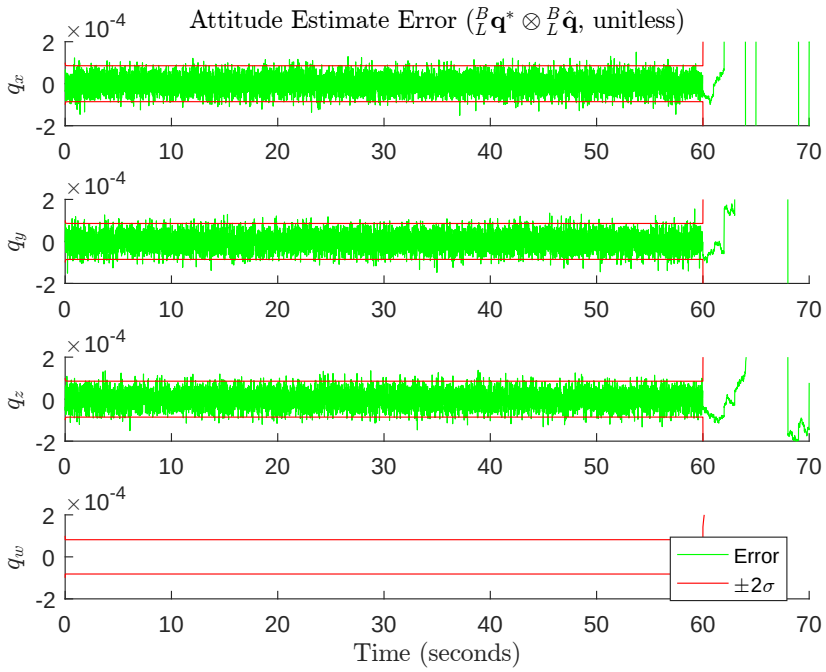

Figure A.8: Random Trajectory - Attitude Estimate Error - Initialization (Quaternion)

A.2 Additional King Air C90 Simulation Data

This section contains supplementary data for Section 4.4. The first set of figures depict the true states produced from the X-Plane simulation. The second set of figures depict the final errors omitted in Section 4.4.2.

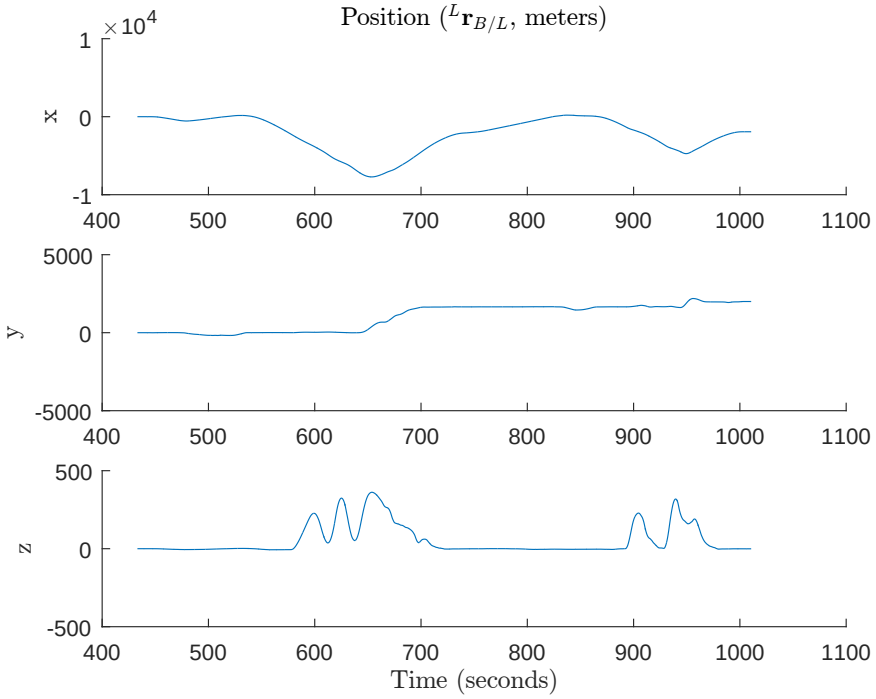

Figure A.9: King Air C90 Simulation - Position

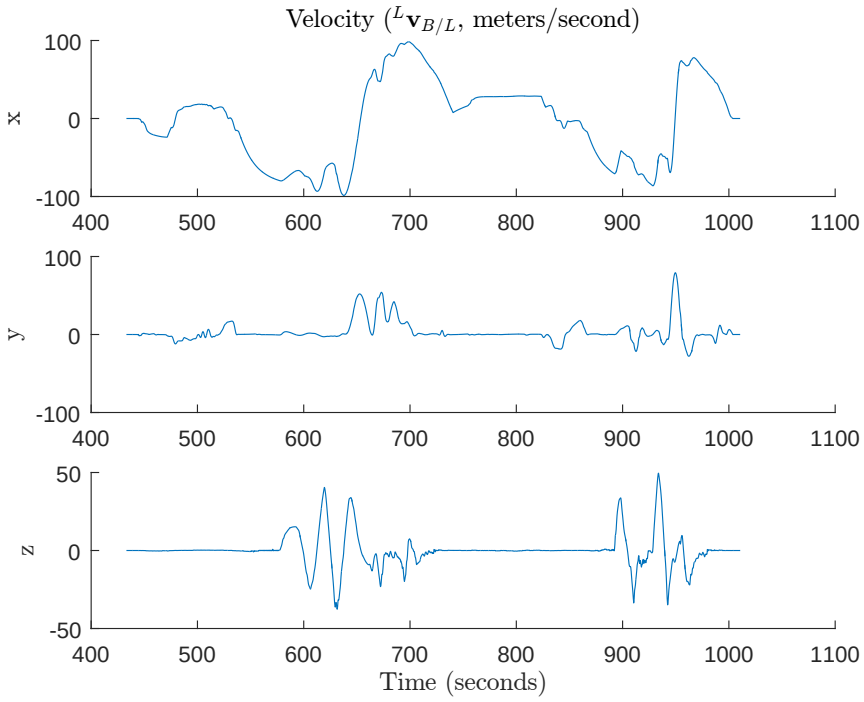

Figure A.10: King Air C90 Simulation - Velocity

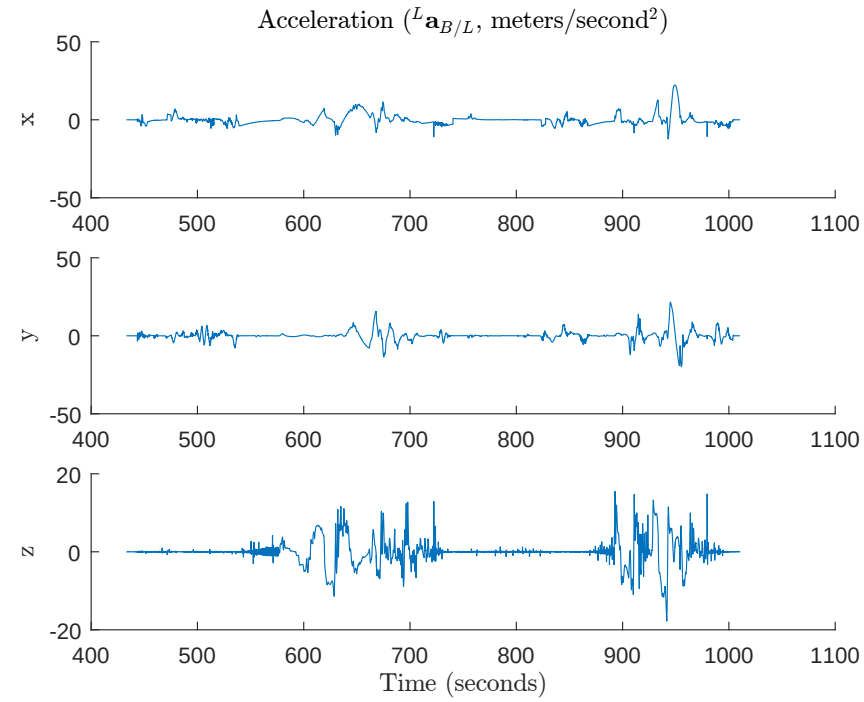

Figure A.11: King Air C90 Simulation - Acceleration

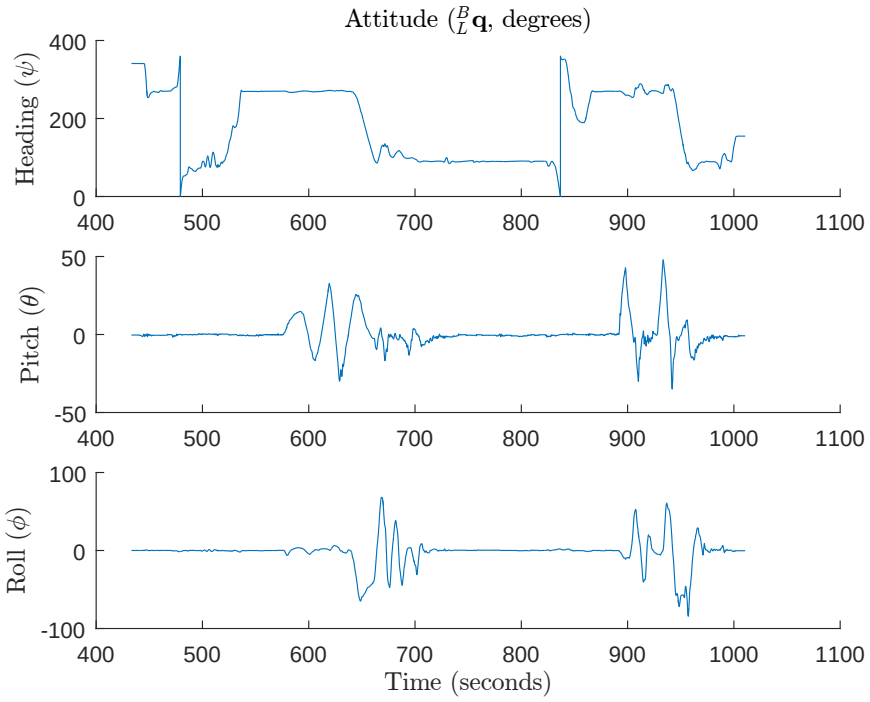

Figure A.12: King Air C90 Simulation - Attitude (Tait-Bryan Angles)

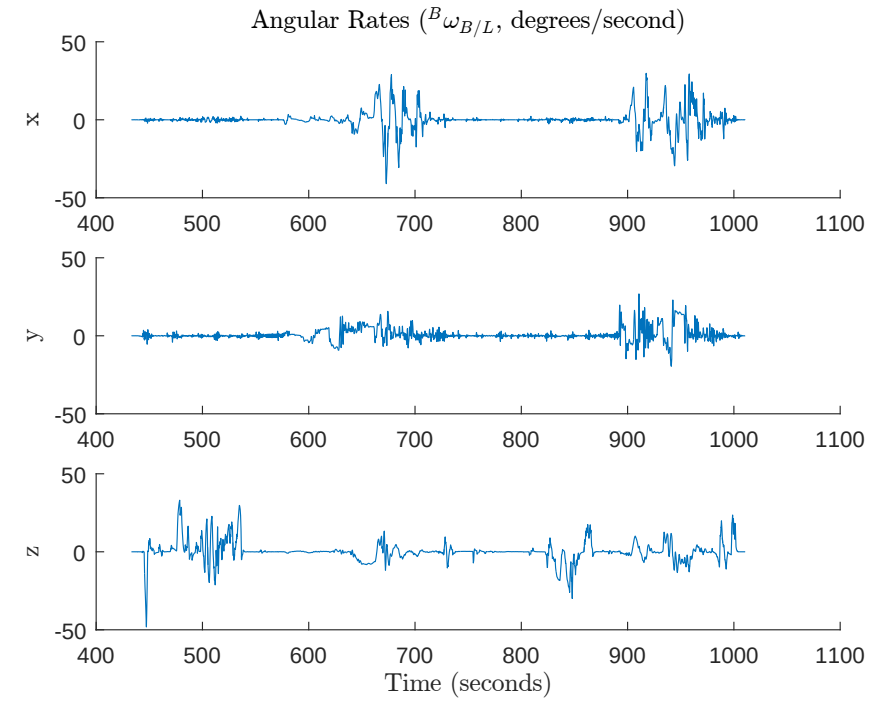

Figure A.13: King Air C90 Simulation - Angular Rates

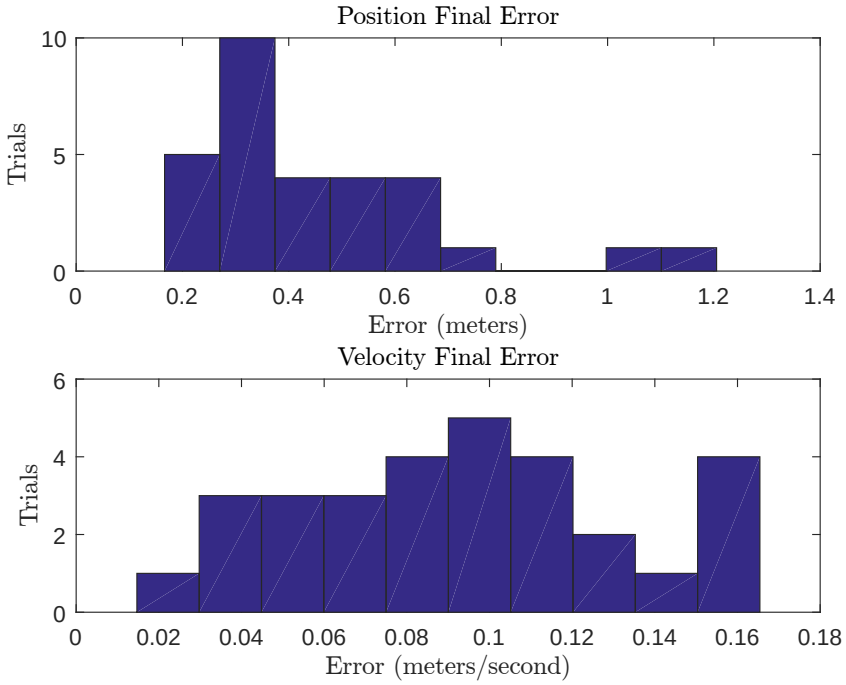

Figure A.14: King Air C90 Simulation - Final Error - Position/Velocity

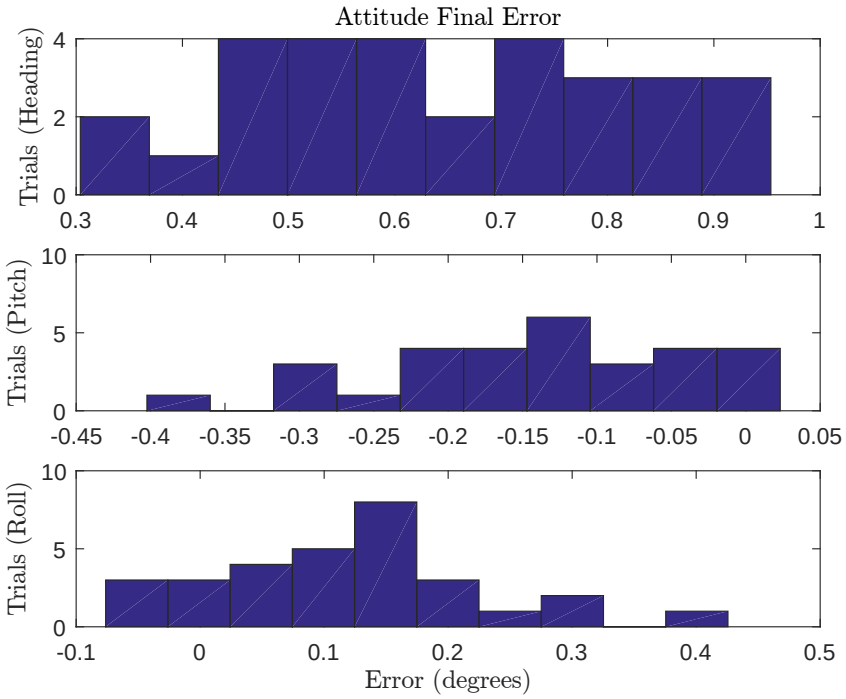

Figure A.15: King Air C90 Simulation - Final Error - Attitude (Tait-Bryan Angles)

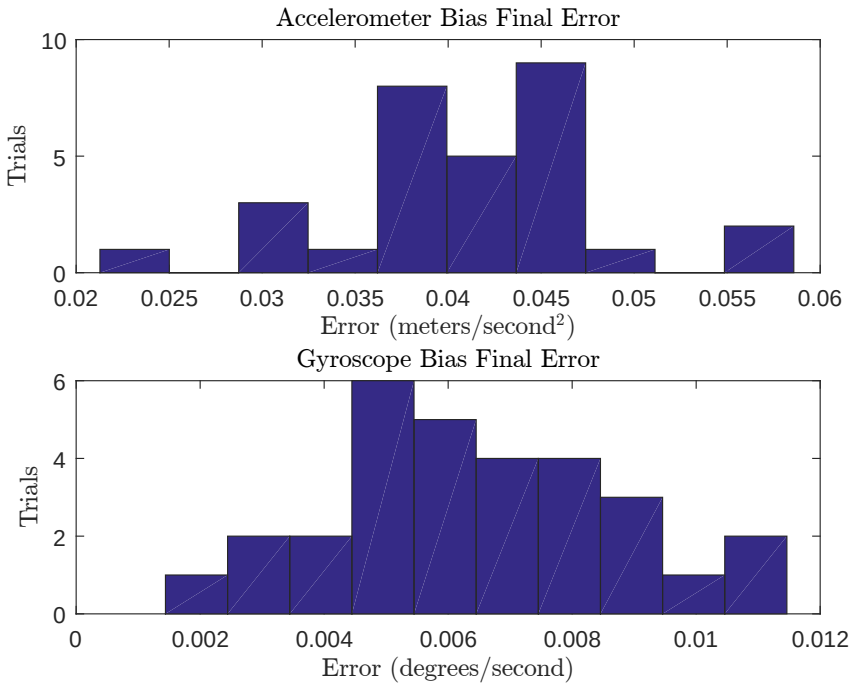

Figure A.16: King Air C90 Simulation - Final Error - Biases

REFERENCES

[1] J. E. D. Williams, *From Sails to Satellites : The Origin and Development of Navigational Science.* Oxford University Press, 1994, ISBN: 9780198563990.

[2] D. Sobel, *Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of His Time*. New York: Walker, 2007, ISBN: 978-0-8027-1529-6.

[3] N. M. Barbour and C. Gibson, "Demonstrating Practical Inertial Navigation: The Beginnings and Beyond," *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013.

[4] P. L. Walter, "The history of the accelerometer: 1920s-1996 - prologue and epilogue, 2006," *Sound & Vibration*, vol. 41, no. 1, pp. 84–92, Jan. 2007, Date revised - 2010-02-01; Last updated - 2011-12-14.

[5] B. McCollum and O. Peters, "A New Electrical Telemeter," *Journal of Chemical Information and Modeling*, vol. 53, p. 160, 1989.

[6] D. Simon, *Optimal State Estimation - Kalman, H-Infinity and Nonlinear Approaches*. 2006, ISBN: 978-0-471-70858-2.

[7] B. Kendal, "The Beginnings of Air Radio Navigation and Communication," *Journal of Navigation*, vol. 64, no. 1, pp. 157–167, 2011.

[8] R. Schroer, "Navigation and Landing [A Century of Powered Flight 1903-2003]," *IEEE Aerospace and Electronic Systems Magazine*, vol. 18, no. 7, pp. 27–36, 2003.

[9] *Radio Navigational Aids*, 117. National Geospatial-Intelligence Agency, 2005.

[10] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems.* Ser. GNSS Technology and Application Series. Artech House, 2013, vol. Second edition, ISBN: 9781608070053.

[11] M. S. Asher, S. J. Stafford, R. J. Bamberger, A. Q. Rogers, D. Scheidt, and R. Chalmers, "Radionavigation Alternatives for US Army Ground Forces in GPS Denied Environments," *International Technical Meeting of The Institute of Navigation*, pp. 508–532, 2011.

[12] M. M. Miller, A. Soloviev, M. Uijt de Haag, M. Veth, J. Raquet, T. J. Klausutis, and J. E. Touma, "Navigation in GPS Denied Environments: Feature-Aided Inertial Systems," DTIC Document, Tech. Rep., 2010.

[13] A. D. Wu, E. N. Johnson, M. Kaess, F. Dellaert, and G. Chowdhary, "Autonomous Flight in GPS-Denied Environments Using Monocular Vision and Inertial Sensors," *Journal of Aerospace Information Systems*, vol. 10, no. 4, pp. 172–186, 2013.

[14] D. M. Sobers, S. Yamaura, and E. N. Johnson, "Laser-Aided Inertial Navigation for Self-Contained Autonomous Indoor Flight," *AIAA Guidance, Navigation, and Control Conference*, pp. 1–56, 2010.

[15] W. Travis, A. T. Simmons, and D. M. Bevly, "Corridor Navigation with a LiDAR / INS Kalman Filter Solution," *IEEE Intelligent Vehicle Symposium Proceedings*, pp. 343–348, 2005.

[16] T. Bailey and H. Durrant-Whyte, "Simultaneous Localization and Mapping (SLAM)," *Update*, vol. 13, pp. 108–117, 2006.

[17] S. Sarkka, "Bayesian Filtering and Smoothing," *Cambridge University Press*, 2013.

[18] C. K. Chui and G. Chen, *Kalman Filtering with Real-Time Applications*. 2009, p. 241, ISBN: 9783540878483.

[19] E. Johnson, *Kalman Filtering*, University Lecture, 2016.

[20] J. Romberg, *The Kalman Filter*, University Lecture, 2016.

[21] G. H. Golub and C. F. V. Loan, *Matrix Computations (Johns Hopkins Studies in the Mathematical Sciences)*. Johns Hopkins University Press, 2012, ISBN: 1421408597.

[22] G. Hemann and H. Hu, "Long-Range GPS-Denied Aerial Inertial Navigation," 2016.

[23] D. J. Higham, "An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations," *SIAM review*, vol. 43, no. 3, pp. 525–546, 2001.

[24] A. Roberts, "Modify the Improved Euler scheme to integrate stochastic differential equations," *ArXiv preprint arXiv:1210.0933*, 2012.

[25] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund, "Particle Filters for Positioning, Navigation, and Tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 425–437, 2002.

[26] R. Van Der Merwe, E. Wan, and S. Julier, "Sigma-Point Kalman Filters for Nonlinear Estimation and Sensor-Fusion: Applications to Integrated Navigation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 5120.

[27] S. J. Julier, J. K. Uhlmann, and H. F. Durrant-Whyte, "A new approach for filtering nonlinear systems," in *American Control Conference, Proceedings of the 1995*, IEEE, vol. 3, 1995, pp. 1628–1632.

[28] S. J. Julier and J. K. Uhlmann, *New Extension of the Kalman filter to Nonlinear Systems*, 1997.

[29] S. Bitzer, *The UKF Exposed: How it works, when it works and when its better to sample*, 2016.

[30] M. Takeno and T. Katayama, "A Numerical Method for Continuous-Discrete Unscented Kalman Filter," *Special Issue of International Journal of Innovative Computing, Information and Control*, vol. 8, no. 3, pp. 2261–2274, 2012.

[31] Carl Carter, "Principles of GPS: A Brief Primer on the Operation of the Global Positioning System," Tech. Rep., 1997.

[32] J. S. Warner and R. G. Johnston, "GPS Spoofing Countermeasures," *Journal of Homeland Security*, 2003.

[33] B. Scherzinger and J. Hutton, "Applanix IN-Fusion(TM) Technology Explained," Tech. Rep., pp. 1–4.

[34] H. Gellersen, P. Lukowicz, M. Beigl, and T. Riedel, "Cooperative Relative Positioning," *IEEE Pervasive Computing*, vol. 9, no. 4, pp. 78–89, 2010.

[35] S. Y. Tan, C. K. Seow, and Y. L. Guan, "A Peer-to-Peer Non-Line-of-Sight Localization System Scheme in GPS-denied Scenarios," DTIC Document, Tech. Rep., 2014.

[36] R Faragher and R Harle, "SmartSLAM - An Efficient Smartphone Indoor Positioning System Exploiting Machine Learning and Opportunistic Sensing," in *ION GNSS*, vol. 13, 2013, pp. 1–14.

[37] A. D. Wu, "Vision-Based Navigation and Mapping for Flight in GPS-Denied Environments," PhD thesis, Georgia Institute of Technology, 2010.

[38] D. Vaman, "TRN History, Trends and the Unused Potential," in *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, 2012, 1A3–1–1A3–16.

[39] R. Collins, *Lecture 7: Correspondence Matching Recall: Derivative of Gaussian Filter*, University Lecture, 2007.

[40] R. Lutwak, "Micro-Technology for Positioning, Navigation, and Timing towards PNT Everywhere and Always," in *Inertial Sensors and Systems (ISISS), 2014 International Symposium on*, IEEE, 2014, pp. 1–4.

[41] J McNeff, "Changing the Game ChangerThe Way Ahead for Military PNT," *Inside GNSS*, vol. 5, no. 8, pp. 44–51, 2010.

[42] M Tanenhaus, D Carhoun, T Geis, E Wan, and A Holland, "Miniature IMU/INS with Optimally Fused Low Drift MEMS Gyro and Accelerometers for Applications in GPS-Denied Environments," in *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*, IEEE, 2012, pp. 259–264.

[43] M. Tanenhaus, T. Geis, D. Carhoun, and A. Holland, "Accurate Real Time Inertial Navigation Device by Application and Processing of Arrays of MEMS Inertial Sensors," in *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, IEEE, 2010, pp. 20–26.

[44] H. Chang, L. Xue, C. Jiang, M. Kraft, and W. Yuan, "Combining Numerous Uncorrelated MEMS Gyroscopes for Accuracy Improvement Based on an Optimal Kalman Filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 11, pp. 3084–3093, 2012.

[45] J. B. Bancroft, "Multiple IMU Integration for Vehicular Navigation," in *Proceedings of ION GNSS*, vol. 1, 2009, pp. 1–13.

[46] J. B. Bancroft and G. Lachapelle, "Data Fusion Algorithms for Multiple Inertial Measurement Units," *Sensors*, vol. 11, no. 7, pp. 6771–6798, 2011.

[47] J. Wang and E. Olson, "High-performance Inertial Measurements Using a Redundant Array of Inexpensive Gyroscopes (RAIG)," in *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2015 IEEE International Conference on*, IEEE, 2015, pp. 71–76.

[48] E. A. Wan and R. Van Der Merwe, "The Unscented Kalman Filter for Nonlinear Estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, IEEE, 2000, pp. 153–158.

[49] Z. Yan, G. She-sheng, and J. Wei-wei, "Robust MP-Augmented UKF Algorithm Integrated Navigation System," 2012.

[50] Z. Jiang, C. Liu, G. Zhang, Y. Wang, C. Huang, and J. Liang, "GPS/INS Integrated Navigation Based on UKF and Simulated Annealing Optimized SVM," in *Vehicular Technology Conference (VTC Fall), 2013 IEEE 78th*, IEEE, 2013, pp. 1–5.

[51] E. Johnson, Personal Conversation, 2017.

[52] S. Kamthe, J. Peters, and M. P. Deisenroth, "Multi-modal Filtering for Non-Linear Estimation," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, IEEE, 2014, pp. 7979–7983.

[53] C. W. Kang and C. G. Park, "A Soft-Failure Detection and Identification Algorithm for the Integrated Navigation System of Lunar Lander," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 230, no. 11, pp. 2023–2035, 2016.

[54] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[55] H. B. Christophersen, R. W. Pickell, J. C. Neidhoefer, A. A. Koller, S. K. Kannan, and E. N. Johnson, "A Compact Guidance, Navigation, and Control System for Unmanned Aerial Vehicles," *Journal of Aerospace Computing, Information, and Communication*, vol. 3, no. 5, pp. 187–213, 2006.

[56] Nvidia, *CUDA Toolkit Documentation - v8.0*, 2017.

[57] B. L. Stevens, F. L. Lewis, and E. N. Johnson, *Aircraft Control and Simulation : Dynamics, Controls Design, and Autonomous Systems*. Hoboken, US: Wiley-Blackwell, 2015.

[58] J. Diebel, "Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[59] VectorNav, "Quaternion Math," Tech. Rep., 2015.

[60] E. W. Weisstein, *Alias Transformation*, 2017.

[61] ——, *Alibi Transformation*, 2017.

[62] R. W. Soutas-Little, D. J. Inman, and D. Balint, *Engineering Mechanics: Dynamics - Computational Edition*. CL Engineering, 2007, ISBN: 0534548857.

[63] J. Sola, "Quaternion Kinematics for the Error-State Kalman Filter," 2017.

[64] J. L. Crassidis and F. L. Markley, "Unscented Filtering for Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 4, pp. 536–542, 2003.

[65] R Chow, "Evaluating Inertial Measurement Units," *Test & Measurement World*, vol. 31, no. 10, pp. 34–37, 2011.

[66] H. Chang, L. Xue, W. Qin, G. Yuan, and W. Yuan, "An Integrated MEMS Gyroscope Array with Higher Accuracy Output," *Sensors*, vol. 8, no. 4, pp. 2886–2899, 2008.

[67] KVH, "Guide to Comparing Gyro and IMU Technologies Micro-Electro-Mechanical Systems and Fiber Optic Gyros," Tech. Rep., 2014, pp. 1–10.

[68] Q. K. Dang and Y. S. Suh, "Sensor Saturation Compensated Smoothing Algorithm for Inertial Sensor Based Motion Tracking," *Sensors*, vol. 14, no. 5, pp. 8167–8188, 2014.

[69] M. Rhudy, Y. Gu, J. Gross, and M. R. Napolitano, "Evaluation of Matrix Square Root Operations for UKF Within a UAV GPS/INS Sensor Fusion Application," *International Journal of Navigation and Observation*, vol. 2011, 2012.

[70] D. H. Titterton, J. L. Weston, A. American Institute of Aeronautics and, and E. Institution of Electrical, *Strapdown Inertial Navigation Technology.* Ser. IET Radar, Sonar, Navigation and Avionics Series Vol. 17. The Institution of Engineering and Technology, 2004, vol. 2nd ed, ISBN: 9780863413582.

[71] M. Liu, Y. Gao, G. Li, X. Guang, and S. Li, "An Improved Alignment Method for the Strapdown Inertial Navigation System (SINS)," *Sensors*, vol. 16, no. 5, p. 621, 2016.

[72] *Systron Donner Inertial SDI500 MEMS Quartz Tactical Inertial Measurement Unit*, 965755, Rev. L, Systron Donner Inertial, Oct. 2016.

[73] *Trimble(R) AG-372 GNSS Receiver User Guide*, Trimble, 2017.

[74] *MEMS Integrated GPS/INS Tactical System*, Systron Donner Inertial, 2017.

[75] *Control System Toolbox: User's Guide*, r2017a, The Mathworks, Inc., 2017.

[76] J. B. Schleppe, "Development of a Real-Time Attitude System using a Quaternion Parameterization and Non-Dedicated GPS Receivers," PhD thesis, University of Calgary, 1997.